File No. 7090-24
Form J28-6260-1

IBM Systems Reference Library

IBM 7090/7094 Programming Systems

COBOL Language: Preliminary Specifications

The COBOL language is an English-like programming language that can be used to solve a wide variety of business and commercial problems. This publication describes the version of COBOL used with the 7090/7094 Data Processing Systems. This version of COBOL is compiled by the COBOL Compiler (IBCBC), which is a component of the 7090/7094 IBJOB Processor.

The reader is assumed to be familiar with the following publications: COBOL, Form F28-8053-2; IBM 709/7090 Input/Output Control System, Form C28-6100-2; IBM 7090/7094 Operating Systems: Basic Monitor (IBSYS), Form C28-6248-0; IBM 7090/7094 Programming Systems: IBJOB Processor, Form C28-6275-0.

This publication is a reprint of Form J28-6260-0 incorporating changes released in Technical Newsletters N28-0040 and N28-0052. A sentence has been added at the end of the section on Basic Operators. While the format has been changed to conform to that of the Systems Reference Library, the original publication and applicable newsletters are not obsoleted.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the content of this publication to:
IBM Corporation, Programming Systems Publications, Dept. D91, PO Box 390, Poughkeepsie, N.Y.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

INTRODUCTION

The material in this bulletin has been prepared by altering the official COBOL-61 manual of the Department of Defense. Many of the elective features and most paragraphs of general commentary have been eliminated, while descriptions of certain adaptations of COBOL to the IBM 7090 and 7094 have been added. In accordance with the requirements of that manual the following is reproduced for the information of the reader:

"This publication is based on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force
Bureau of Standards, Department of Commerce
David Taylor Model Basin, Bureau of Ships, U. S. Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
Burroughs Corporation
International Business Machines Corporation
Radio Corporation of America
Sylvania Electric Products, Inc.
Univac Division of Sperry-Rand Corporation

In addition to the organizations listed above, the following other organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
Bendix Corporation, Computer Division
Control Data Corporation
DuPont Corporation
General Electric Company
General Motors Corporation
Lockheed Aircraft Corporation
National Cash Register Company
Philco Corporation
Standard Oil Company (N.J.)
United States Steel Corporation

"This COBOL-61 manual is the result of contributions made by all of the above-mentioned organizations. No warranty, expressed or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

"It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

"Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures and the methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

"The authors and copyright holders of the copyrighted material used herein: FLOW-MATIC (Trade-mark of Sperry-Rand Corporation), Programming for the UNIVAC® I and II, Data Automation Systems © 1958, 1959, Sperry-Rand Corporation; IBM Commercial Translator, Form No. F28-8013, copyrighted 1959 by IBM; FACT, DSI 27A5260-2760, copyrighted 1960 by Minneapolis-Honeywell, have specifically authorized the use of this material, in whole or in part, in the COBOL specifications. Such authorization extends to the reproduction and use of COBOL specifications in programming manuals or similar publications.

"Any organization interested in reproducing the COBOL report and initial specifications in whole or in part, using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention COBOL in acknowledgement of the source, but need not quote this entire section."

NOTATION

This section details the notation conventions used in describing required and elective COBOL syntax.

1. All upper case words which are underlined are required when the functions of which they are a part are used. An error will occur at compilation time if the underlined words are absent or incorrectly spelled.

2. All upper case words which are not underlined are used for readability only. They may be present or absent in the source program.

3. All lower case words represent generic terms which must be supplied by the user.

4. Material enclosed in braces { } indicates that a choice from the contents must be made.

5. Material enclosed in square brackets [ ] represents an option, and may be included or omitted by the user.

6. Notes will elaborate on the formats and specify any restrictions.

7. In cases where many choices are available, some separations into numbered options have been made.

8. When two or more nouns may be written in a series, commas are shown as connectives. Where a comma is shown in the formats, it may be omitted or replaced by either AND or , AND.

Part I: BASIC FACTS

This section defines the IBM COBOL character set and describes the types and formation of COBOL names. It also includes special topics such as punctuation, subscripting, and name qualification.

CHARACTER SET

Complete IBM COBOL Character Set

The complete IBM COBOL character set consists of the following 48 characters:

0 - 9
A - Z
(blank) Blank or space
\+ Plus sign
\- Minus sign or hyphen
* Asterisk
/ Stroke (virgule or slash)
= Equal sign
$ Dollar sign
, Comma
. Period or decimal point
' Quotation mark
(Left parenthesis
) Right parenthesis

Characters Used For Words

The character set for words consists of the following 37 characters:

0 - 9
A - Z
\- (hyphen or minus)

Note that blank or space is not an allowable character for a word, but is used to separate words. Where a blank or space is employed, more than one may be used, except where restricted by the Reference Format (see Part VI).

Characters Used For Punctuation

The punctuation characters consist of the following:

' Quotation mark
(Left parenthesis
) Right parenthesis
(blank) Blank or space

.	Period
,	Comma

Characters Used In Formulas

+	Addition
-	Subtraction (hyphen)
*	Multiplication
**	Exponentiation
/	Division
=	Equality

Character Used In Relations

=	Equal to

Additional Characters Used In Editing

$	Dollar sign
*	Check protection symbol
,	Comma
.	Actual decimal point

WORDS AND GOVERNING RULES

Definition of Words

A word is composed of a combination of not more than 30 characters chosen from the following set of 37 characters:

0 - 9
A - Z
- (hyphen)

The following rules govern the definition of a word:

1. A word is ended by a space or by either a period, right parenthesis, or comma, followed by a space.

2. A word may not begin or end with a hyphen.

3. A literal constitutes an exception to the preceding rules (see page 7).

Use of Punctuation Characters with Words

The following rules govern the use of punctuation characters:

1. A period or comma, when used, must immediately follow a word, but a period or comma in turn, must be followed by a space.

2. A left parenthesis must not be followed immediately by a space, and a right parenthesis must not be immediately preceded by a space.

3. A beginning quotation mark must not be followed by a space, or an ending quotation mark preceded by a space, unless the spaces are desired in the literal.

Types of Words

Nouns

A noun is a single word which is one of the following types:

Data-name
Condition-name
Procedure-name
Literal
Figurative constant
Special register name
Special name

A noun may contain hyphens for readability purposes. For example, QUANITY-ON-HAND, and STOCK-NUMBER are legitimate nouns, whereas, STOCKNUMBER- is not, since a word must not end with a hyphen. (Labels, tags, field-names, operation numbers, and other such terms used in other languages, are considered nouns in the COBOL language.)

The rules which define the types of nouns are as follows:

1. Data-names. A data-name is a word with at least one alphabetic character, which designates any data specified in the data description, or the area which contains the data referred to.

2. Condition-names. A condition-name is the name assigned to a specific value, within the complete set of values that a data-name. may assume. The data-name itself is called a conditional variable, and those values it may assume are referred to by condition-names. The condition-name must contain at least one alphabetic character.

Each condition-name must be unique, or be made unique through qualification. A conditional variable may be used as a qualifier for any of its condition-names.

If a condition-name is to be equated to the status of a hardware device, it is defined in the SPECIAL-NAMES paragraph of the Environment Division.

The difference between this kind of condition-name and a Data Division condition-name is automatically handled by the compiler.

3. Procedure-names. A procedure-name is either a paragraph-name or a section-name. Procedure-names permit one procedure to refer to others. A procedure-name may be composed solely of numeric characters. However, two numeric procedure-names are equivalent only if they are composed of the same number of numeric digits and have the same numeric value. Thus, 0023 is not equivalent to 23.

4. Literals. A literal is an item of data, integral to the text, which is completely defined by its own identity rather than by Data Division clauses. A literal may belong to one of two classes: non-numeric (alphabetic or alphanumeric) or numeric. Non-numeric literals must be bounded by quotation marks; numeric literals must not.

   A non-numeric literal is defined as a literal which is composed of up to 120 of any allowable characters except the quotation mark. All spaces which are enclosed in the quotation marks are included as spaces in the literal.

   A numeric literal is defined as one which is composed only of characters chosen from the numerals 0 through 9, the plus or minus sign, and the decimal point. The rules for formation of numeric literals are:

   a. A numeric literal may contain only one sign character and/or one decimal point.

   b. The literal must contain at least one and not more than 18 digits.

   c. The sign in the literal must appear as the left-most character. If the literal is unsigned, it is considered to be positive.

   d. The decimal point may appear anywhere within the literal except as the right-most character, and is treated as an implied decimal point. If the literal contains no decimal point, it is considered to be an integer.

(See page 62 for discussion of floating point literals.)

If a literal conforms to the rules for formation of numeric literals, but it is enclosed in quotation marks, it is considered as a non-numeric literal and will be treated as such by the compiler. For example, -125.65 is not the same as '-125.65'.

Examples of non-numeric literals are:

```
'EXAMINE CLOCK NUMBER'
'PAGE 144 MISSING'
'-125.65'
```

Examples of numeric literals are:

```
1506798
-12572.6
+256.75
1435.89
```

5. Figurative Constants. Certain values have been assigned fixed data-names. These items with the fixed data-names are called "figurative constants." These names, when used as figurative constants, must not be bounded by quotation marks. The singular and plural forms of figurative constants are equivalent, and may be used interchangeably. If the names are bounded by quotation marks they will be considered as non-numeric literals. The fixed data-names and their meanings are as follows:

ZERO ZEROS ZEROES	Represents the value 0.
SPACE SPACES	Represents one or more blanks or spaces.
HIGH-VALUE HIGH-VALUES	Normally represents one or more left parentheses, but may represent one or more 9s if the COLLATE-COMMERCIAL option has been specified.
LOW-VALUE LOW-VALUES	Normally represents one or more zeros, but represents one or more spaces if the COLLATE-COMMERCIAL option is used.
QUOTE QUOTES	Represents the character '. Note that the use of the word QUOTE to represent the character ' at object time is not equivalent to the use of the symbol ' to bound a literal.

ALL 'literal'	Represents one or more occurences of the single character (bounded by quotation marks) comprising the literal. An alternative form for "literal" is any figurative constant (not bounded by quotation marks) other than the ALL 'literal' type, e.g., ALL SPACES.

Figurative constants generate a string of homogeneous information whose length is determined by the compiler, based upon context. When the length is not deducible from context, a single character is generated. The figurative constants may be used in the Procedure and Data Divisions.

Examples of the above are:

a. MOVE ALL '4' TO COUNT-FIELD, where COUNT-FIELD has been described as having six characters, results in 444444.

b. From the statement MOVE SPACES TO TITLE-BOUNDARY, the compiler will create coding which puts as many space characters into the item TITLE-BOUNDARY as are necessary to fill the item.

c. DISPLAY QUOTE, 'NAME', QUOTE results in ' NAME ' .

d. MOVE QUOTE TO AREA-A, where AREA-A has been described as having five characters, results in """".

6. Special Register. TALLY is the name of a special register (USAGE COMPUTATIONAL, SYNCHRONIZED RIGHT) whose length is equivalent to a five decimal digit integer. It may be used to hold intermediate results during the execution of a program.

7. Special Names. Special names provide a means of relating hardware with problem-oriented names, and the status of hardware switches with condition-names.

Verbs

A verb is a single word which appears in the Procedure Division, and designates an action: ADD, MOVE, GO TO, etc.

Reserved Words

Reserved words are used for syntactical purposes and may not be used as nouns or verbs. There are three types:

1. Connectives. Connectives are words used to:

   a. Denote the presence of a qualifier: OF, IN.

   b. Form compound conditionals: AND, OR, AND NOT, OR NOT; these are called logical connectives.

2. Optional Words. Optional words have been defined and used to improve the readability of the language. Within each format, upper case words which are not underlined are designated as optional. The presence or absence of each optional word within the format for which it is shown does not alter the compiler's translation. However, misspelling of an optional word or its replacement by another word of any kind is not permitted.

3. Key Words. Key words are of three types:

   a. Verbs: ADD, READ, ENTER, etc.

   b. Required words, which appear in formats in various divisions of the language and which are needed to complete the meaning of certain verbs or entries: TO, OMITTED, MEMORY, etc.

   c. Words not shown in any format, but which have a specific functional meaning: NEGATIVE, SECTION, TALLY, etc.

Qualifiers

Every name used in a COBOL source program must be unique, either because no other name has the identical spelling, or because the name exists within a hierarchy of names (so that the name can be made unique by mentioning one or more of the higher levels of the hierarchy). The higher levels are called qualifiers when used in this way, and the process is called qualification. Enough qualification must be mentioned to make the name unique, but it is not necessary to mention all levels of the hierarchy unless they are needed to make the name unique. A file-name is the highest level qualifier available for a data-name. A section-name is the highest and the only qualifier available for a paragraph-name. Thus, file-names and section-names must be unique in themselves and cannot be qualified.

Qualification in COBOL is performed by appending one or more prepositional phrases, using IN or OF. The choice between IN or OF is based on readability, since they are logically equivalent. Nouns must appear in ascending order of hierarchy with either of the words IN or OF separating them. The qualifiers are considered part of the name. Thus, whenever a data item or procedure paragraph is referred to, any necessary qualifiers must be written as part of the name.

The following additional rules must be observed in using qualification:

1. A qualifier must be of a higher level and within the same hierarchy as the name it is qualifying.

2. The same name may not appear at two levels in a hierarchy so that it would appear to qualify itself.

3. If a data-name or condition-name is assigned to more than one data item in a program, it must be qualified in all references to it in the Procedure Division and the Environment Division, and in all references to it in the defining clauses in the Data Division.

4. A paragraph-name must not be duplicated within the same section and can be qualified by only a section-name. When it is, the word SECTION must not appear. A paragraph-name need not be qualified when referred to from within the same section. Subscripts and conditional variables, as well as procedure- and data-names may be made unique by qualification where necessary or desirable.

5. A data-name cannot be subscripted when it is being used as a qualifier.

6. The highest level qualifier must be unique. Furthermore, each qualifying name must be unique at its own level within the hierarchy of the immediately higher qualifier.

7. A name can be qualified even though it does not need qualification. The use of more names for qualification than are actually required for uniqueness is permitted. If there is more than one combination of qualifiers which will ensure uniqueness, then any set can be used.

8. The name of a conditional variable can be used as a qualifier for any of its condition-names.

Subscripts

The technique of subscripting is most commonly used for table-handling functions. The ability to refer to individual elements (of a table or list) which have not been assigned individual data-names is provided by using subscripts; the ability to refer to the entire table or list is provided by using the name of the table or list.

A subscript is an integer whose value determines which element is being referred to within a table (or list) of like elements. The subscript may be represented either by a literal which is an integer or by a data-name which has an integral value.

When the subscript is represented by a data-name, the data-name must be described by a Record Description entry in the Data Division. In both cases, i.e., whether the subscript is represented by a literal or a data-name, the subscript is enclosed in parentheses, and appears immediately after the terminal space of the name of the element. Tables are often defined so that more than one level of subscripting is required to locate an element within them. A maximum of three levels of subscripting is permitted by COBOL. Multilevel subscripts are always written from left to right in the order: major, intermediate, minor. In this case, the subscripts are shown in a single pair of parentheses, and separated by commas. For example:

```
RATE (REGION, STATE, CITY)
RATE (3, STATE, CITY)
RATE (3, 5, 6)
```

All of the above would refer to a particular rate in a three-dimensional table of rates.

A subscript value of 1 denotes the first element of a list, a value of 2, the second element, etc. A subscript of (1,2) denotes the second element within the first repeated element of the table. A table with its main element appearing ten times, its intermediate element appearing five times within each of the major elements, and the minor element appearing three times within each of the intermediate elements, is considered a three-dimensional table. The last element of such a table is referred to by the use of the subscript (10, 5, 3).

If a data item is repeated, i.e., involves the OCCURS clause at its own or higher level, then the name of this item must be subscripted whenever it is used. Furthermore, a data-name can be subscripted only if the data item is repeated (see the OCCURS clause in the Data Division for details).

The same data-name used as a subscript may be associated with items of different sizes. Within a given program, the same data-name may appear as one of several subscripts with one item, and as the only subscript with another item.

Regardless of the above rules, a data-name must <u>not</u> be subscripted under any of the following conditions:

1. where the data-name is being used as a subscript.

2. where the data-name is being used as a qualifier.

3. when the data-name appears in the defining clauses in the Data Division.

There are several correct ways of expressing subscripted data-names. For example, if a data item named A occurs five times and contains a data item named B which occurs four times in each A, and each B, in turn, contains a data item C which occurs twice in each B, then the following expressions are all correct references to the last C, i.e., to the second C in the fourth B in the fifth A:

```
C IN B IN A  (5, 4, 2)
C IN B  (5, 4, 2)
C IN A  (5, 4, 2)
C  (5, 4, 2)
```

The following forms of expression are incorrect:

```
C  (5, 4, 2)  IN B IN A
C  (2)  IN B  (4)  IN A  (5)
C  (4, 2)  IN A  (5)
C  (2)  IN B  (5, 4)
```

Series Separators

When two or more nouns are written in a series, words or characters may be used as separators between the nouns. The use of such separators in a series of nouns is optional. The separators which may be used are:

```
, AND
,
AND
```

PART II. IDENTIFICATION DIVISION

The Identification Division of a source program is used to identify the program and the output of a compilation. In addition, it may include the date that the program was written and any other information which may be considered desirable.

ORGANIZATION

The format of the Identification Division is relatively brief and straightforward, as shown below:

```
{ IDENTIFICATION DIVISION. }
{ ID DIVISION.             }

  PROGRAM-ID. program-name.

[ AUTHOR. author-name.                                  ]
[ INSTALLATION. any sentence or group of sentences.     ]
[ DATE-WRITTEN. any sentence or group of sentences.     ]
[ DATE-COMPILED. any sentence or group of sentences.    ]
[ SECURITY. any sentence or group of sentences.         ]
[ REMARKS. any sentence or group of sentences.          ]
```

Note: Unless one of the other division names appears in the A-margin of the program sheet, the compiler will treat what follows the Identification Division as part of the Identification Division until it reaches the end of the source program.

ADDITIONAL INFORMATION

PROGRAM-ID

Function

To give the name by which a program is identified.

PROGRAM-ID. program-name.

Notes:

1. The program-name is a word (conforming to the rules for a word) that is used to identify compiler output.

2. The PROGRAM-ID will appear at the beginning of the COBOL assembly output listing as a comments card.

Optional Entries

1. The remaining six entries may or may not be included at the user's option. Any entries that are included will be produced with the assembled output listing, as comments.

2. The date supplied by the monitor program with the assembled output listing will not be affected by the DATE-COMPILED entry.

Part III: ENVIRONMENT DIVISION

The Environment Division is used to specify those aspects of the total data processing problem which are dependent upon the physical characteristics of the specific computer. It provides a linkage between the logical concepts of data and records, and the physical aspects of the files on which they are stored.

GENERAL DESCRIPTION

Organization

The Environment Division has been divided into two sections: Configuration and Input-Output.

The Configuration Section, which deals with the overall specifications of computers, is divided into three paragraphs: the SOURCE-COMPUTER, which identifies the computer on which the COBOL Compiler is to be run; the OBJECT-COMPUTER, which identifies the computer on which the program produced by the COBOL Compiler is to be run; and SPECIAL-NAMES, which relate the actual names of the hardware used by the COBOL Compiler to the names used in the program.

The Input-Output Section deals with the definition of the external media and information needed to create the most efficient transmission and handling of data between the media and the object program. This section is divided into two paragraphs: the I-O-CONTROL, which defines re-run procedures; the FILE-CONTROL, which names and associates the files with the external media.

Structure

The following is a general outline of the sections and paragraphs under the Environment Division.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  computer name...
OBJECT-COMPUTER.  computer name...
SPECIAL-NAMES.  hardware-name IS ...
INPUT-OUTPUT SECTION.
FILE-CONTROL.  SELECT...
I-O-CONTROL.
```

CONFIGURATION SECTION

SOURCE-COMPUTER

Function

To identify the computer upon which the program is to be compiled.

SOURCE-COMPUTER $\left\{ \begin{matrix} \underline{\text{IBM7090}} \\ \underline{\text{IBM7094}} \end{matrix} \right\}$

Notes:

1. Whether the IBM 7090 or the IBM 7094 is used as the source computer, the following list designates the hardware used by the compiler:

   a. Main Frame
      (1) 32,768 words of core storage.
      (2) Data Channel trap.
      (3) Three index registers - 1, 2, and 4.
   b. Auxiliary (devices referred to are IB Basic Monitor units)
      (1) System tape - SYSLB1.
      (2) Input and Output units - SYSIN1, SYSOU1, [SYSPP1].
      (3) Utility units - Four System Utility units.

2. The compiler always operates under control of a supervisor, which is a subsystem fuctioning under control of IBSYS.

OBJECT-COMPUTER

Function

To describe the characteristics of the computer upon which the program is to be run.

```
OBJECT-COMPUTER. { IBM7090 }
                 { IBM7094 }

                 [{ MEMORY SIZE    32768 WORDS }]
                  [{ ADDRESS 0 THRU 32767       }]

                 [COLLATE-COMMERCIAL]
```

Notes:

1. The following list designates the hardware used either by the object deck loader or the object program:

   a. Main Frame
      (1) 32,768 words of core storage.
      (2) Data Channel trap.
      (3) Three index registers (1, 2, and 4) if IBM 7090; seven index registers (1, 2, 3, 4, 5, 6, and 7) if IBM 7094.

   b. Auxiliary (devices referred to are IBSYS units)
      (1) System Tape - SYSLB1
      (2) Input and Output units - SYSIN1, SYSOU1.
      (3) Utility units

2. COLLATE-COMMERICAL specifies that the commercial collating sequence is to be used in all procedural alphanumeric comparisons in the object program. In the absence of COLLATE-COMMERCIAL, the object program will use the standard 7090/7094 collating sequence for alphanumeric comparisons.

3. The object deck loader and the object program always operate under control of a supervisor which is a subsystem functioning under control of IBSYS.

4. The optional MEMORY and ADDRESS clauses are treated by the compiler as non-functional remarks.

SPECIAL-NAMES

Function

To provide a means of relating hardware with mnemonic-names, and the status of hardware switches with condition-names.

```
SPECIAL-NAMES.  { KEY-S  }                    [ IS mnemonic-name-1 ]
                { KEY-1  }
                { KEY-2  }
                {  .     }
                {  .     }
                { KEY-35 }

                { , ON STATUS IS condition-name-1                    }
                {                                                    }
                { , OFF STATUS IS condition-name-2                   }
                {                                                    }
                { , ON STATUS IS condition-name-3,                   }
                {     OFF STATUS IS condition-name-4                 }

                [ entire set of clauses may repeat                   ]
                [   with the preceding format . . . . .              ]
```

Notes:

1. This paragraph is not required if mnemonic-names and condition-names are not used in the Procedure Division.

2. This paragraph is used to assign condition-names to the ON and/or OFF status of console switches. The console switches which may be tested are the 36 console entry keys S, 1-35. By definition, a switch is ON if it is down and OFF if it is up.

3. In this paragraph the IS mnemonic-name-1 clause is used for readability purposes only, as this mnemonic-name cannot be referred to by a legitimate conditional expression.

4. Setting of all console switches is performed by the machine operator at object-time. An installation may have its own restrictions as to which (if any) console switches may be used by object-programs.

INPUT-OUTPUT SECTION

FILE-CONTROL

Function

To name each file referred to in the Procedure Division and to specify hardware assignments.

Option 1.

```
FILE-CONTROL.  SELECT [OPTIONAL] file-name-1
    [RENAMING file-name-2]
    ASSIGN TO { 1 TAPE-UNIT
              | 1 TAPE-UNIT FOR MULTIPLE REEL
              | 2 TAPE-UNITS FOR MULTIPLE REEL
              | CARD-READER
              | PRINTER
              | CARD-PUNCH }
    [SELECT...]
```

Option 2.

```
FILE-CONTROL.  SELECT [OPTIONAL] file-name-1
[RENAMING file-name-2]
    ASSIGN TO { symbolic-tape-unit-name-1
              | symbolic-tape-unit-name-2
                FOR MULTIPLE REEL
              | symbolic-tape-unit-name-3
              | symbolic-tape-unit-name-4
                FOR MULTIPLE REEL
              | symbolic-card-unit-name-1
              | system-unit-name-1
              | system-unit-name-2
                FOR MULTIPLE REEL
              | system-unit-name-3, system-unit-name-4
                FOR MULTIPLE REEL
              | NONE }
[SELECT....].
```

Notes:

1. The beginning of the information for each file-name-1 is identified by the key word SELECT.

2. The key word OPTIONAL is required for input files which will not necessarily be present each time the object program is to be run.

3. The file-name-1 names a file. It is this SELECT file-name which is referred to in the Procedure Division OPEN, CLOSE, and READ clauses; and it is also this file-name which is referred to in the Environment Division I-O-CONTROL paragraph. There must be a SELECT entry for every file name which is referred to in the program. Each SELECT file entry must specify the Data Division File Description entry which gives the structure of the file. This is accomplished in one of two ways:

   a. When the RENAMING clause is not used, file-name-1 must be the name of a Data Division File Description entry, and that name must be unique within the program.

   b. When the RENAMING clause is used, file-name-1 must be a unique name which does not appear as the name of a Data Division File Description entry. The Data Division File Description which is used for file-name-1 is the same description that is used for file-name-2. The file-name-2 SELECT entry must specify which Data Division File Description is to be used.

4. RENAMING a file implies the sharing of a single File Description entry and the storage allocated for that entry, but does not allow the file names to be referred to interchangeably; that is, the file names are not synonyms. Two files (SELECT entries) which share the same FD entry must not be in OPEN status at the same time.

5. If a file is to be processed both as an input file and as an output file, there must be two SELECT entries for that file.

6. The only difference between Option 1 and Option 2 is in the form of the ASSIGN clause. Option 1 is used to assign files to one of four general hardware categories: TAPE-UNIT [S], CARD-READER, CARD-PUNCH or PRINTER. When this form is used the programmer has no control over which specific physical unit within a given hardware category is to be assigned to the file, i.e., when the hardware category is TAPE-UNITS, any one of the tape units on any one of the channels may be assigned to the file. The actual physical unit assigned to a file is printed on the on-line printer just prior to the execution of the object program. Option 2 is used to make a more specific assignment of units to files based upon knowledge of the relationships between files in the source program, and a knowledge of more sophisticated unit assignment techniques. The definition of the Option 2 forms of the ASSIGN clause is given below.

7. A given file may be assigned to one or two but not more than two physical tape units. When a file uses two physical tape units, the units are used alternately for successive reels.

8. The MULTIPLE REEL option must be used when the file being described may exist on two or more different reels of magnetic tape. This option must be specified when two tape units are used for the file. It must also be specified when a single tape unit is to be used for a file but the file is contained on two or more reels of tape.

9. Unit Assignment. The assignment of physical input/output units to each file is performed by IBLDR prior to execution of the object program. This assignment is accomplished in IBLDR by interpreting the Unit 1 and Unit 2 fields on the $FILE cards which precede the binary deck. The ASSIGN clause in COBOL provides a means at compilation time to specify the contents of the Unit 1 and Unit 2 fields on all $FILE cards. When Option 2 is used, the unit names which may be assigned to a file are as follows:

   a. Symbolic tape units. The programmer may assign files to symbolic tape unit names, and these symbolic tape units are in turn equated to actual available physical tape units by IBLDR. The notation used in explaining the options which can be used in choosing symbolic tape unit names to be assigned to a file is as follows:

Symbol Shown	Significance of Symbol Shown
X	Denotes one of the real channels A,B,...,H.
P	Denotes a symbolic (unspecified physical) channel S, T, ..., Z.
k	Denotes one of the unit numbers 1, ..., 9,0.
M	Denotes an IBM 729 II or IV Magnetic Tape Unit.
I	Denotes an inter-system (unspecified physical) channel J,K,...,Q.

Symbolic inter-system units may be specified as follows:

Symbolic Name	Interpretation Given to the Name by IBLDR
I	Inter-system channel.
IM	Inter-system channel and model.
I(k)	Inter-system channel and relative unit.
I(k)M	Inter-system channel, relative unit, and model.

The symbolic names (other than inter-system units) may be chosen from any of the following set of names:

Symbolic Name	Interpretation Given to the Name by IBLDR
M	Any available tape unit of this model type is to be assigned to the file.
X	Any available tape unit on this physical channel is to be assigned to the file.
P	All files in the program having this symbolic channel designation are to be assigned to the same channel.
X(k)	The (k)th available tape unit on the specified channel is to be assigned to the file. Note that the parentheses are required.
PM	Any available tape unit of this model type on the specified symbolic channel is to be assigned to the file.
P(k)M	An available tape unit on the symbolic channel having this model number is to be assigned to the file. The (k) in this usage indicates the order of preference for the channel so that if the number of available tape units on the channel is less than the total requested for the channel, those with lower numbers are to be assigned to the same channel. Note that the parentheses are required.

b. Symbolic Unit Record Equipment Names.
The following symbolic unit record equipment names may be assigned to a file: X has the same meaning as in symbolic tape units:

Symbolic Name	Interpretation Given to the Name by IBLDR
RDX	The card reader on channel X is to be assigned to the file.
PRX	The printer on channel X is to be assigned to the file.
PUX	The card punch on channel X is to be assigned to the file.

c. System Units. IBSYS units may be assigned to files by using the following names:

System Name	Interpretation Given to the Name by IBLDR
SYSIN1	The current system input unit is to be assigned to the file. Note: The physical unit which is attached to this file is controlled by the machine operator and may be a card reader or a tape unit.
SYSOU1	The current system print output unit is to be assigned to the file. Note: The physical unit which is attached to this file may be a printer or a tape unit.
SYSPPi	The current system peripheral punch unit is to be assigned to this file. Note: The physical unit which is attached to this file may be a card punch or a tape unit.
SYSUTk	System utility tape k (k=1, 2, 3, or 4) is to be assigned to the file.
SYSLBk	System library tape k (k=1, 2, 3, or 4) is to be assigned to the file.
SYSCRD	The system card reader is to be assigned to the file.
SYSPRT	The system printer is to be assigned to the file.
SYSPCH	The system card punch is to be assigned to the file.

d. NONE. Files which are designated as OPTIONAL files may have the unit NONE assigned to the file. When NONE is placed in the Unit 1 field of a $FILE card, that file is said to be absent for that particular run and no input activity will be applied to it.

Within certain limitations, it is permissible to change the contents of the Unit 1 and Unit 2 fields of the $FILE cards after the program has been compiled. One restriction on changing unit assignments after compilation is that files which are assigned to tape media only in the source program (this includes files assigned to NONE in the source program) can not be changed to card media; files which are assigned to card media only in the source program (this includes the printer) can not be changed to tape media. Files which are assigned to either card or tape media in the source program (SYSIN1, SYSOU1, and SYSPP1) may be changed to any other tape or card assignment for a given run of the object program. A file which is assigned to a system unit must conform to the physical and logical characteristics inherent in the system unit. The table on the following pages summarizes the relationships between the permissible forms of the ASSIGN TO clause and the contents of associated fields in the $FILE cards which are produced by the compiler.

File Assignment Table

Form of ASSIGN TO clause	Contents of Associated $FILE Card Fields		
	Unit 1 Field	Unit 2 Field	Multireel Field
1 TAPE-UNIT	blank	blank	blank
1 TAPE-UNIT MULTIPLE REEL	blank	blank	MULTIREEL
2 TAPE-UNITS MULTIPLE REEL	blank	blank	MULTIREEL
CARD-READER	CRD	blank	blank
CARD-PUNCH	PCH	blank	blank
PRINTER	PRT	blank	blank
symbolic-tape-unit-name-1	symbolic-tape-unit-name-1	blank	blank
Example: A	A	blank	blank
symbolic-tape-unit-name-2 MULTIPLE REEL	symbolic-tape-unit-name-2	blank	MULTIREEL
Example: TIV MULTIPLE REEL	TIV	blank	MULTIREEL
symbolic-tape-unit-name-3 symbolic-tape-unit-name-4 FOR MULTIPLE REEL	symbolic-tape-unit-name-3	symbolic-tape-unit-name-4	MULTIREEL
Example: C(3), C(4) MULTIPLE REEL	C(3)	C(4)	MULTIREEL
symbolic-card-unit-name-1	symbolic-card-unit-name-1	blank	blank
Example: RDA	RDA	blank	blank
system-unit-name-1	abbreviated system-unit-name-1	blank or may be automatically filled in	
*Example: SYSIN1	IN1	IN2	MULTIREEL
system-unit-name-2 MULTIPLE REEL	abbreviated system-unit-name-2	blank or may be filled in automatically	MULTIREEL
Example: SYSUT1 MULTIPLE REEL	UT1	blank	MULTIREEL
system-unit-name-3, system-unit-name-4	abbreviated system-unit-name-3	abbreviated system-unit-name-4	MULTIREEL
Example: SYSUT2, SYSUT3 FOR MULTIPLE REEL	UT2	UT3	MULTIREEL
NONE	NONE	blank	blank

*Note: In the example, this particular file automatically takes on the MULTIREEL characteristic of the system unit. This is also true for SYSOU1 and SYSPP1 which are system MULTIREEL files.

I-O-CONTROL

Function

To specify rerun procedures.

```
I-O-CONTROL.

RERUN { ON CHECKPOINT-UNIT                    }
      { EVERY END OF REEL OF file-name-1       }
      { EVERY END OF REEL OF output-file-name-2 }

[RERUN. . .].
```

Notes:

1. This paragraph is required only when rerun is desired.

2. RERUN may be used to specify that memory dumps are to be written under one of the following conditions:

   a. At each reel switch of the specified file (whether input or output), a memory dump is to be written on the unit which is provided by the compiler as the standard checkpoint-unit.

   b. At each reel switch of a labeled output file, a memory dump is to be written following the header label on the new reel of that output file.

3. Within the INPUT-OUTPUT section, the I-O-CONTROL paragraph must not precede the FILE-CONTROL paragraph.

Part IV: DATA DIVISION

The Data Division of a source program is used to specify the format and organization of the data to be processed by the compiled program.

GENERAL DESCRIPTION

Overall Approach

Data to be processed falls into three categories, (a) that which is contained in files and enters or leaves the internal memory of the computer from a specified area or areas, (b) that which is developed internally and placed into intermediate or working storage, and (c) constants which are defined by the user. (Figurative constants and literals used in procedure statements are not listed in the Data Division.) Tables may fall into any of the above categories.

The approach taken in defining file information is to distinguish between the physical aspects of the file (the File Description) and the conceptual characteristics of the data contained therein (the Record Description). Physical aspects refer to the mode in which the file is recorded, the grouping of logical records within the physical limitations of the file-medium, the means by which the file can be identified, etc. Conceptual characteristics refer to the explicit definition of each logical entity within the file itself.

For purposes of processing, the contents of a file are divided into logical records. By definition, a logical record is the set of information defined by a Record Description in the Data Division. It is important to note that several logical records may occupy one block (a physical tape record). However, logical records may not extend across physical records.

The concept of a logical record is not restricted to file data, but is carried over into the definition of working storages and constants. Thus, working storages and constants may be grouped into logical entities and defined by a series of Record Description entries.

Organization

The Data Division, which constitutes one of the divisions involved in a problem definition, is subdivided according to types of data. It consists of a File Section, a Working-Storage Section, and a Constant Section, written in that order.

The File Section contains two elements, descriptions of files and descriptions of records. The File Description entry represents the highest level of organization in the File Section.

The Working-Storage and Constant Sections consist solely of Record Descriptions and unrelated Record Description entries.

Structure

The Data Division of the source program begins with the header:

DATA DIVISION.

Each of the three sections begins with the appropriate section-name, followed by the word SECTION and a period, e.g., FILE SECTION., or WORKING-STORAGE SECTION.. When a section is not required in the definition of the problem, its name need not appear in the source program.

The sections themselves consist of a series of related and unrelated entries. This is different from the paragraph, sentence, statement structure which characterizes the Procedure, Environment, and Identification Divisions.

An entry consists of a level indicator, a data-name, and a series of independent clauses which may be separated by commas. The clauses may be written in any sequence, except when the entry format specifies otherwise. The entry itself is terminated by a period. A File Description consists of a single entry, whereas a Record Description consists of one or more entries.

The data-name (or the key word FILLER) which appears immediately after the level indicator is called the "subject" of the entry. A data-name (or FILLER) may appear as the subject of more than one entry. When a duplicated data-name is referred to in one of the defining clauses of an entry, it must be qualified whenever used.

FILE DESCRIPTION ENTRY

General Description

A File Description entry contains information pertaining to the physical aspects of a file. In general, it may include the following: the manner in which the data is recorded on the file, the size of the logical and physical records, the values of specific label items contained in the file, and the names of the data records which comprise the file.

The listing of data record names in a File Description entry serves as a cross-reference between the file and the records in the file.

Entry Formats

General Notes

A File Description entry consists of a level indicator, a file-name, and a series of independent clauses which define the physical and logical characteristics of the file. The mnemonic level indicator FD is used to identify the start of a File Description entry, and distinguishes this entry from those associated with a Record Description.

Specific Formats

The individual clause formats are arranged in an alphabetic order in this bulletin, but the clauses are shown in the recommended order in the "Complete Entry."

COMPLETE ENTRY

Function

To furnish information concerning the physical structure, identification and record descriptions pertaining to a given file.

```
FD file-name [, RECORDING MODE IS mode]
     [, BLOCK CONTAINS [integer-1 TO] integer-2 {RECORD [S]     }]
                                                {CHARACTER [S]}
     , RECORD CONTAINS [integer-3 TO] integer-4 CHARACTER [S]]
     , LABEL {RECORD IS   } {STANDARD}
             {RECORDS ARE } {OMITTED }
     [, VALUE OF label-data-name-1 IS literal-1
          [, label-data-name-2 IS ...]]
     , DATA {RECORD IS   } data-name-1 [, data-name-2 ...]
            {RECORDS ARE }
```

---

1. The level indicator FD identifies the beginning of the File Description and precedes the file-name.

2. The clauses which follow the name of the file are optional in many cases. For further details, see the individual explanations for each clause.

3. The File Description entry is terminated by a period.

4. All commas are optional in the File Description.

5. Those entries which require more than a single line are continued on subsequent lines with the same left margin for each line, i.e., with each subsequent line starting under the first character in the file-name (see the Reference Format for the Data Division).

BLOCK SIZE

Function

To specify the size of a physical record (block).

```
[,BLOCK CONTAINS [integer-1 TO] integer-2 { RECORD [S]    }]
                                          { CHARACTER [S] }
```

Notes:

1. Blocking of logical records is permitted only when one of the following conditions is true:

   a. There is only one record-name in the DATA RECORD clause for the file. In this case the record may be fixed or variable length.

   b. There are several record-names in the DATA RECORD clause for the file, but all of the records have the same fixed length.

2. Blocking of files which are assigned to card equipment is not permitted.

3. This clause is not required when the file is composed of physical records each of which contains only one complete logical record. The form BLOCK CONTAINS 1 RECORD is implied in this case.

4. Logical records are not allowed to extend across physical records, i.e., in no case can a physical record contain a fractional part of a logical record.

5. Integer-1 and integer-2 must be numeric literals with positive integral values.

6. The RECORDS form of the clause is used when it is desired to specify the size of the block in terms of the number of logical records in the block. The CHARACTERS form of the clause is used when it is desired to specify the size of the block by describing the number of physical computer characters (six times the number of computer words) in the block. When specifying the number of computer characters, careful consideration must be given to data items which are either SYNCHRONIZED or COMPUTATIONAL.

7. The word CHARACTERS within the BLOCK clause is an optional word. Whenever the key word RECORDS is not specifically written in the BLOCK clause, integer-1 and integer-2 represent CHARACTERS.

8. The table below specifies the interpretations given to the four permissible forms of the BLOCK clause. The abbreviation FIX in the table is used to specify a file which contains logical records of a given fixed length. VAR in the table is used to specify a file which contains logical records of varying length. This is the case where there is only one record name in the DATA RECORD clause in the FD entry, but that record name is designated to be of variable length through the use of an OCCURS... DEPENDING ON... clause in the data description of one or more data-items within that record. Buffer-size in the table is the size of core storage areas reserved by the compiler for either receiving physical records from input files or for forming physical records to be written upon output files.

   RECLTH in the table is an abbreviation for the length of a logical record. The units used in expressing RECLTH are computer words, i.e., one sixth the number of characters in the logical record. RECLTH is preceded by the abbreviation MAX or MIN to denote the maximum length and the minimum length of a logical record, respectively. Note that the last block of a file may be shorter and may not necessarily comply with a specification which calls for an exact number of logical records per physical block. The physical blocks are never padded by the compiler to conform with a specification which calls for an exact number of logical records per physical block.

   If a file containing logical records of varying length is assigned to a binary mode tape, an extra word in front of each logical record is assumed when reading or is automatically supplied when writing. MAXRECLTH, therefore, equals logical record length plus one. This extra word convention provides compatibility with 7090 Sort and with one type of 7090 Commercial Translator file.

Table of Permissible Forms of BLOCK Clause

Form of BLOCK Clause	File Type	Buffer-size Assigned by Compiler	Interpretation Given to Clause
a. integer-2 RECORDS	(1)FIX	(integer-2) * (RECLTH)	If the file is input, each block must contain exactly integer-2 logical records. If output the compiler will write exactly integer-2 logical records in each block.
	(2)VAR	(integer-2)* (MAXRECLTH)	Same as a(1)
b. integer-1 TO integer-2 RECORDS	(1)FIX	(integer-2)* (RECLTH)	If input, each block may contain integer-1 to integer-2 logical records. If output, the compiler will write up to integer-2 logical records into each block.
	(2)VAR	(integer-2)* (MAXRECLTH)	If input, each block may contain integer-1 to integer-2 logical records. If output, the compiler will fill each block until integer-2 logical records have been placed in the block.
c. integer-2 CHARACTERS	(1)FIX	(integer-2)/6	The number of logical records in the physical block for both input and output files is calculated as (integer-2)/6/(RECLTH). If the result is not an integer, an error is indicated. Processing at object time is the same as in a(1) using this calculated blocking factor as the exact number of logical records per physical record.
	(2)VAR	(integer-2)/6	This is given an interpretation as if the clause read BLOCK CONTAINS 6 TO integer-2 CHARACTERS. Interpretation, therefore, is the same as in d(2) when 6 is substituted for integer-1.

Table of Permissible Forms of BLOCK Clause - (Contd.)

Form of BLOCK Clause	File Type	Buffer-size Assigned by Compiler	Interpretation Given to Clause
d. integer-1 TO integer-2 CHARACTERS	(1)FIX	(integer-2)/6	The number of logical records in each physical block may range from (integer-1)/6 / (RECLTH) to (integer-2)/6 / (RECLTH). Both integer-1 and integer-2 must be exactly divisible by 6.
	(2)VAR	(integer-2)/6	(integer-1)/6 and (integer-2)/6 represent the minimum and maximum blocksize respectively. In reading an input file, it is assumed that the block contains an integer number of records (an error will result at object time if this is not true) and the integer must be within the range of (integer-1)/6/ (MAXRECLTH) to (integer-2)/6/ (MINRECLTH). If the file is an output file, the compiler will ensure that logical records do not extend across blocks. Each block is truncated in processing when the next logical record to be written out might not fit into the current block being formed.

DATA RECORDS

Function

To cross-reference the description of data records with their associated file.

,DATA { RECORDS ARE / RECORD IS } data-name-1 [, data-name-2 . . .]

Notes:

1. This clause is required in every File Description entry.

2. The presence of more than one data-name indicates that the file contains more than one type of data record. These records may be of differing sizes, different formats, etc. The order in which they are listed is not significant.

3. Conceptually, all data records within a file share the same area. This is in no way altered by the presence of more than one type of data record within the file.

4. Both data-name-1 and data-name-2 must have 01 level numbers. Subscripting of these data-names is not permitted, and qualification is not necessary since they are implicitly qualified by the file-name of this entry.

LABEL RECORDS

Function

To indicate the presence or absence of standard labels.

```
                  { RECORDS ARE }        {STANDARD}
, LABEL           { RECORD IS   }        {OMITTED }
```

Notes:

1. The Input/Output Control System contains the mechanism to process files which contain labels with a standard format. If a file contains labels which have a non-standard format or if the file contains no labels at all, the OMITTED form must be used, and processing of any non-standard labels must then be accomplished by the READ and WRITE verbs.

2. There is only one storage area reserved for all standard labels, whether input or output, and all labels are processed in that area.

   This area and all names within the area are defined by the compiler, i.e., the data description of the area is contained within the compiler and must not be repeated in the source programs.

3. Standard labels are of two types, header labels and trailer labels. The various items within a label are processed in different ways depending upon which type label is being read.

4. Since there is only one label area shared by all files, any references to items within that area must not be qualified by a file name, i.e., each name in the label area is unique.

5. The standard label area is processed by the IOCS for any of one of the following purposes:

   a. To check an input file header label. A header label occurs at the beginning of the file and at the beginning of every reel of the file. At the time this type of label is processed by IOCS, the items FILE-SERIAL-NUMBER, REEL-SEQUENCE-NUMBER, and FILE-IDENTIFICATION are checked for the values designated in the respective VALUE OF clauses under the FD entry for the file.

   b. To check an input file trailer label. A trailer label occurs at the end of the file and at the end of every reel of the file. If the value of the item LABEL-IDENTIFIER is '1EORb', the label is an END-OF-REEL label; if the value is '1EOFb', the label is an END-OF-FILE label. At this time the BLOCK-COUNT item (this item applies to trailer labels only) is checked against the number of blocks actually read by the IOCS.

   c. To check the header label on a tape to be used in order to insure that the tape may be written upon. This is accomplished by determining if the RETENTION-PERIOD in the header label of that tape has expired.

   d. To prepare an output file header label. At the time this label is formed by the IOCS the items RETENTION-PERIOD, REEL-SEQUENCE-NUMBER, FILE-IDENTIFICATION and FILE-SERIAL-NUMBER are formed from the information supplied in the respective VALUE OF clauses under the FD entry for the file.

   e. To prepare an output file trailer label. At this time the LABEL-IDENTIFIER item is filled with the proper END-OF-REEL or END-OF-FILE indication, and the BLOCK-COUNT item is also filled in with the actual number of physical blocks that have been written on the reel.

RECORD SIZE

Function

To specify the size of data records.

[, RECORD CONTAINS [integer-3 TO] integer-4 CHARACTERS]

Notes:

1. Integer-3 and integer-4 must be numeric literals with positive integral values.

2. The size of each data record is completely defined within the Record Description entries; therefore this clause is never required. When present, however, the following notes apply:

   a. In this clause, CHARACTERS refers to the number of computer characters which the record will occupy. Therefore, when specifying the number of computer characters, careful consideration must be given to items which are either synchronized or computational.

   b. Integer-4 may not be used by itself unless all the data records in the file have the same size. In this case integer-4 represents the exact number of characters in the data record. If integer-3 and integer-4 are both shown, they refer to the minimum number of characters in the smallest size of data record and the maximum number of characters in the largest size of data record, respectively.

RECORDING MODE

Function

To specify the external format of a file.

```
[, RECORDING MODE IS  {BCD   } [{LOW  }  DENSITY]]
                      {BINARY} [{HIGH }          ]
```

Notes:

1. The BCD mode should be specified for any file assigned exclusively to card equipment. If the unit assigned to the file may be either a tape or card unit, as is the case with some system units, the recording mode must agree with the system unit recording mode.

2. If information is recorded on the magnetic tape just as it is found in core storage (except for check bits), the tape is said to be a binary tape or to have been written in the binary recording mode. Through the use of auxiliary units, alphanumeric information may be recorded on or read from a magnetic tape independently of the computer. Magnetic tapes prepared or used by this auxiliary equipment employ a special coding system known as binary-coded-decimal. Tapes which are prepared using the binary-coded-decimal system are said to be BCD tapes, i.e., the RECORDING MODE IS BCD. Tapes recorded in the BCD mode cannot contain

data items whose USAGE IS COMPUTATIONAL. Therefore, all data items recorded in the BCD mode must be described as USAGE IS DISPLAY.

3. Each tape unit is capable of writing characters on magnetic tapes at two densities, LOW or HIGH. The density of a tape refers to the number of characters that can be written on a given area of tape. The more characters that can be written, the higher the character density. Some auxiliary equipment requires information to be recorded in LOW DENSITY. However, if the tape is to be completely processed on the 7090/94, the HIGH DENSITY option should be used to minimize the tape area required to contain a given number of characters.

4. If the clause is not used, RECORDING MODE will be BCD, HIGH DENSITY.

VALUE

Function

To specify the contents of particular items in the standard labels associated with a file.

```
[, VALUE OF label-data-name-1  IS literal-1 [label-data-name-2 IS ...]]
```

Notes:

1. Each label-data-name is a data-name item and must be an item in the standard label area (see LABEL RECORDS).

2. When label-data-name is alphanumeric, literal-1 must be a non-numeric literal (enclosed within quotation marks). When label-data-name is numeric, literal-1 must not be enclosed in quotation marks, and must consist only of numerals.

3. Files may have label-data-names conforming to the following lists:

   a. Input files.

Label-data-name	Contents of literal
FILE-IDENTIFICATION	Ten or less alphanumeric characters which identify the file.

Label-data-name	Contents of Literal
FILE-SERIAL-NUMBER	Five or less alphanumeric characters. This is the REEL-SERIAL-NUMBER of the file. The REEL-SERIAL-NUMBER of a reel of tape is the number on the external casing of that reel.
REEL-SEQUENCE-NUMBER	Four or less numerals. This is the number of the reel within a given file, i.e., the first reel of a file is reel 1, the second is reel 2. The value specified in the VALUE OF clause is checked against the value in the label of the first reel of the file which is processed. Every time a reel switch occurs the value is increased by one, and the new value is checked against the value in the label of the new reel.

Each of the above items may or may not be specified for a standard label input file. If the item is not specified, it is not checked when the file is processed.

b. Output files.

Label-data-name	Contents of literal
FILE-IDENTIFICATION	Ten or less alphanumeric characters which identify the file.
RETENTION-PERIOD	Four or less numerals. This is the number of days that the file is to be saved after its date of creation. The creation date for the file is supplied by the input/output system at the time the file is created.
FILE-SERIAL-NUMBER	Five or less alphanumeric characters. This item has the same definition as in input (see above).

Label-data-name	Contents of literal
REEL-SEQUENCE-NUMBER	Four or less numerals. This item has the same definition as above.

Each of the preceding items may or may not be specified for a standard label output file. The IOCS applies the following procedures to these items:

1. If FILE-IDENTIFICATION is specified, the non-numeric literal is placed in the labels created. If it is not specified, the first ten characters of the file name are placed in the labels created.

2. If RETENTION-PERIOD is specified, that value is placed in the labels created. If it is not specified, the value placed in the labels created will be zero.

3. The value of FILE-SERIAL-NUMBER is placed in the label only if the REEL-SEQUENCE-NUMBER value specified is greater than one. If the REEL-SEQUENCE-NUMBER is not specified as greater than one, the value used for FILE-SERIAL-NUMBER in creating a new label will come from the label of the reel chosen to be the next one used.

4. If REEL-SEQUENCE-NUMBER is specified, that value will be placed in the label of the first reel of the file. Each reel after the first reel will get a REEL-SEQUENCE-NUMBER one greater than the previous reel. If REEL-SEQUENCE-NUMBER is not specified, the value used for the first reel will be one.

RECORD DESCRIPTION

General Description

Elements of a Detailed Data Description

A Detailed Data Description consists of a set of entries. Each entry defines the characteristics of a particular unit of data. With minor exceptions, each entry is capable of completely defining a unit of data. Because the COBOL Detailed Data Descriptions involve a hierarchical structure, the contents of an entry may vary considerably, depending upon whether or not it is followed by subordinate entries.

In defining the lowest level or subdivision of data, the following information may be required:

a. A level-number which shows the relationship between this and other units of data.

b. A data-name.

c. The size in terms of the number of digits or characters.

d. The usage of the data.

e. The number of consecutive occurrences (OCCURS) of elements in a table or list.

f. The class or type of data, (ALPHABETIC, NUMERIC, or ALPHANUMERIC).

g. The type of sign.

h. Location of an actual or an assumed decimal point.

i. Location of editing symbols such as dollar signs and commas.

j. Synchronization of the data (SYNCHRONIZED).

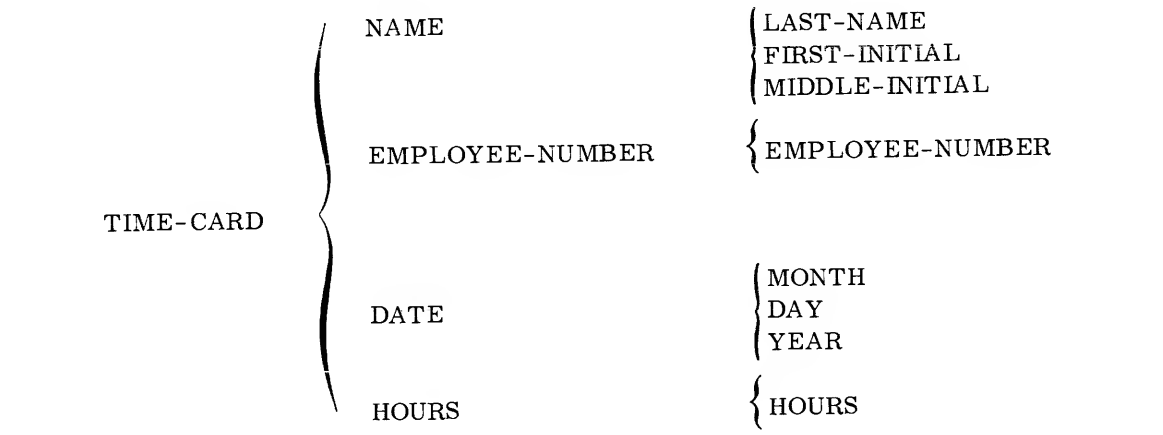
k. Special editing requirements such as zero suppression and check protection.

l. Initial value (VALUE) of a working-storage item or the fixed value (VALUE) of a constant.

An entry which defines a unit of data must not be contradicted by a subordinate entry. Thus, once the class is defined, it applies to all subordinate entries and need not be re-specified. However, when the class is defined as alphanumeric, subordinate entries may particularize the class by specifying alphabetic or numeric. If the class has been defined as either alphabetic or numeric, subordinate entries may not change the class.

Concept of Levels

A level concept is inherent in the structure of a logical record. It arises in a natural way from the need to specify subdivisions of a record for the purpose of data reference. Once a subdivision has been specified, it may be further subdivided to permit more detailed data references. For example, a weekly TIME-CARD record might be divided into four major items: NAME, EMPLOYEE-NUMBER, DATE, and HOURS, with more specific information on DATE and NAME as follows:



The most basic subdivisions of a record (those not further subdivided), are called "elementary items". Consequently, a record is said to consist of a sequence of elementary items.

Often it is desirable to refer to a set of elementary items, particularly with the MOVE verb. For this reason, elementary items may be combined into "groups", each group consisting of a sequence of one or more elementary items. Groups, in turn, may be combined into aggregations of two or more groups, etc. The term "item", in this bulletin, denotes either an elementary item or a group.

A system of level-numbers is employed in COBOL to show the organization of elementary items and groups. Level-numbers start at 01 for records, since records are the most inclusive groups possible. Less inclusive groups are assigned higher (not necessarily successive) level-numbers not greater in value than 49. (Note: There are "special" level-numbers, 77 and 88, discussed below, which are exceptions to this rule.) Separate entries are written in the source program for each level.

Using the Time-Card example above, a skeleton source program listing, showing the use of level-numbers to indicate the hierarchical structure of the data, might appear as follows:

```
01      TIME-CARD
        04 NAME
              06 LAST-NAME
              06 FIRST-INITIAL
              06 MIDDLE-INITIAL
        04 EMPLOYEE-NUMBER
        04 DATE
        05 MONTH
        05 DAY
        05 YEAR
        04 HOURS
```

For the sake of simplicity, only the level-number and data-name of each entry have been given in the above example. A complete description would have included information on size, class, usage, etc.

A group includes all groups and elementary items described under it until a level-number less than or equal to the level-number of that group is encountered. Thus, in the above example, HOURS is not a part of the group called DATE. MONTH, DAY, and YEAR are a part of the group called DATE, because they are described immediately under it and have a higher level-number.

It should be noted that an elementary item may belong to more than one group. In the previous example, the elementary item called YEAR belongs to the group called DATE, and also to the group called TIME-CARD.

The level-number of an entry (elementary item or group) immediately following the last elementary item of a previous group, must be that of one of the groups to which that elementary item belongs. Application of this rule to the previous example restricts the level-number of HOURS to either 1 or 4, since these are the level-numbers of the groups containing YEAR. Only the level of 04 permits HOURS to be a sub-organization of TIME-CARD.

Two types of data exist for which there is no true concept of level:

1. Constants and working-storage items which are not members of any hierarchy. They are called independent constants and independent working-storage items, and have been assigned the special level-number 77.

2. Entries which specify condition-names to be associated with a particular value of an item, which do not themselves introduce data, and which have been assigned the special level-number 88.

COMPLETE ENTRY SKELETON

Function

To specify the characteristics of a particular item of data.

```
level-number { data-name } [, REDEFINES ...] [, SIZE ...] [, USAGE ...]
             { FILLER    }

[, OCCURS...] [, SIGNED] [, SYNCHRONIZED...] [, POINT ...]
[, CLASS ...] [, PICTURE...]
[, editing clauses ...]                      [, VALUE...]
```

Notes:

1. For a detailed explanation of the Reference Format used in the Data Division, see Part VI of this manual.

2. Those clauses which begin with SIGNED, SYNCHRONIZED, POINT, and PICTURE, as well as the editing clauses, must not be specified except at the elementary item level.

3. The clauses may be written in any order with one exception: REDEFINES, when used, must immediately follow the data-name.

4. All commas are optional in the Record Description entry.

5. The Record Description entry is terminated by a period.

ENTRY FORMAT

General

When the format shown in the Complete Entry Skeleton is applied to specific items of data it will be limited by the nature of the data being described. This section describes the allowable format for the description of each possible type of data-item (except those with level 88).

Conventions Used In The Following Data Entry Formats

1. Any clauses which are not shown in a format are specifically forbidden in that format.

2. When a clause which has more than one possible form is shown in its skeletal form, all options are legal. When such a clause is written out any option not shown is forbidden. For example, [CLASS...] allows all options while [CLASS IS NUMERIC] forbids the use of the AN and ALPHABETIC options.

3. Clauses (and options within clauses) which are enclosed in boxes are legal, but undesirable.

4. In the description of some types of data the appearance of certain clauses is mandatory. Such clauses are written in the formats with no brackets.

5. Parenthesized numbers refer to notes which follow.

6. In the PICTURE clause,
   a. form-a is a combination of As, Xs and 9s but not all 9s
   b. form-b is a legal combination of 9 V P and S
   c. form-c is a legal combination of 9 V P , . + - Z $ B CR DB 0 or * with at least one character not 9 V or P
   d. form-d is $\left\{\begin{matrix}+\\-\end{matrix}\right\}$ 9(m) $\left[\left\{\begin{matrix}V\\.\end{matrix}\right\}\right]$ 9(n) E $\left\{\begin{matrix}+\\-\end{matrix}\right\}$ 99

Specific Formats

Group Items

level-number $\left\{\begin{matrix}\text{data-name}\\ \underline{\text{FILLER}}\end{matrix}\right\}$ [REDEFINES...] [OCCURS...]

[SIZE...] [USAGE...] [CLASS...] .

Elementary Items

Alphanumeric (non-report) Items and Alphabetic Items

level-number $\left\{\begin{matrix}\text{data-name}\\ \underline{\text{FILLER}}\end{matrix}\right\}$ [REDEFINES...] [OCCURS...]

[SIZE... (1)] [SYNCHRONIZED...] [PICTURE IS form-a]

[USAGE IS DISPLAY] [VALUE IS non-numeric-literal]

$\left[\text{CLASS IS } \left\{\begin{matrix}\underline{\text{ALPHANUMERIC}}\\ \underline{\text{ALPHABETIC}}\ (2)\end{matrix}\right\}\right]$

Report Items

level-number { data-name / FILLER } [REDEFINES. . .] [OCCURS. . .]

[SYNCHRONIZED. . .] [POINT. . . (1)] [editing clauses (1)]

[SIZE. . . (1)] [CLASS IS ALPHANUMERIC] [USAGE IS DISPLAY]

[PICTURE IS { form-b (3) / form-c }] [VALUE IS numeric-literal]

External-Decimal Items

level-number { data-name / FILLER } [REDEFINES. . .] [OCCURS. . .]

[SYNCHRONIZED. . .] [POINT. . . (1)] [SIZE. . . (1)]

[SIGNED. . . (1)] [USAGE IS DISPLAY] [PICTURE IS form-b]

[VALUE IS numeric-literal] [CLASS IS { ALPHANUMERIC (4) / NUMERIC }]

Internal-Decimal Items

level-number { data-name / FILLER } [REDEFINES. . .] [OCCURS. . .]

[POINT. . . (1)] [SIZE. . . (1)] USAGE IS COMPUTATIONAL

[SIGNED. . . (6)] [PICTURE IS form-b] [VALUE IS numeric-literal]

[CLASS IS NUMERIC] [SYNCHRONIZED { LEFT (5) / RIGHT }]

Floating-Point Items

```
level-number { data-name } [ REDEFINES. . . ] [ OCCURS. . . ]
             { FILLER    }

USAGE IS { COMPUTATIONAL-1 } [ SYNCHRONIZED. . . (7) ]
         { COMPUTATIONAL-2 }

[ VALUE IS { numeric-literal         } ] [ CLASS IS NUMERIC ]
           { floating-point-literal  }
```

Scientific-Decimal Items

```
level-number { data-name } [ REDEFINES. . . ] [ OCCURS. . . ]
             { FILLER    }

PICTURE IS form-d [ SYNCHRONIZED. . . ]

[ CLASS IS { ALPHANUMERIC (4) } ] [ USAGE IS DISPLAY ]
           { NUMERIC          }
```

Notes:

1. Certain clauses are regarded as mutually exclusive. These are:

   a. PICTURE and POINT
   b. PICTURE and SIGNED
   c. PICTURE and SIZE
   d. PICTURE and all editing clauses (except BLANK WHEN ZERO).

   In case of such a contradictory occurrence, PICTURE dominates.

2. This stipulates that this field is alphabetic. Therefore, if the PICTURE clause is used, it must contain all As.

3. This form of the PICTURE clause requires, in addition, an editing clause to define a report field.

4. Since this is a numeric item at an elementary level, it is more accurate to describe it as CLASS IS NUMERIC, although it is not incorrect to describe it as AN.

5. The most efficient form for a NUMERIC COMPUTATIONAL item is SYNCHRONIZED RIGHT.

6. It is redundant to specify a sign for an internal-decimal item.

7. A SYNCHRONIZED clause has no effect when used to describe a floating-point item, since such an item always occupies one or two full words (depending on precision).

CLASS

Function

To indicate the type of data being described.

```
[, CLASS IS { ALPHABETIC   }  ]
            { NUMERIC      }
            { ALPHANUMERIC }
            { AN           }
```

Notes:

1. AN is an acceptable abbreviation for ALPHANUMERIC.

2. If AN is specified, it implies that USAGE is DISPLAY.

3. The CLASS clause can be written at any level. If the CLASS clause is written at a group level, it applies to each elementary item in the group. The class of an item cannot contradict the class of a group to which the item belongs. If a group item is specified as AN, all items subordinate to that group are, by implication, DISPLAY items. ALPHABETIC or NUMERIC items within an ALPHANUMERIC group are not considered contradictory.

4. A data item can be called NUMERIC only if it is composed solely of characters chosen from the numerals 0-9, and an operational sign. If there is no sign associated with the data, the data is considered positive. If the data is numeric and no assumed decimal point is indicated, the data is considered to be an integer.

5. ALPHABETIC describes data which contains any combination of the twenty-six letters of the English alphabet and the space. No other characters, and no numerals can be used.

6. ALPHANUMERIC describes data which may contain any character in the valid character set. Thus, data which is ALPHABETIC or NUMERIC is also ALPHANUMERIC.

7. If a PICTURE is given, the CLASS clause is unnecessary. If both are used, however, the class shown by the PICTURE must not contradict the CLASS clause of an elementary item, or of a group to which the item belongs.

DATA-NAME FILLER

Function

To specify the name of the data being described, or to specify a portion of the logical record to which no reference is made.

```
{ data-name }
{ FILLER    }
```

Notes:

1. A data-name or the key word FILLER must be the first word following the level-number in each Record Description entry and must not be qualified or subscripted.

2. Qualification of a data-name is automatically provided through higher level data-names and file-names. Thus a data-name need not be unique within or between Record Descriptions provided a higher level data-name or a file-name can be used for qualification.

3. Under no circumstances may a FILLER item be referred to directly.

EDITING CLAUSES

Function

To specify suppression of non-significant zeros and commas, or to specify floating dollar signs or check protection.

```
[ , { ZERO SUPPRESS      } [ LEAVING integer PLACE [S] ] ]
    { CHECK PROTECT      }
    { FLOAT DOLLAR SIGN  }
                           [ BLANK WHEN ZERO ]
```

Notes:

1. The editing clauses can be specified only at the elementary item level.

2. The rules for editing, as shown in the MOVE verb, specify that data items are moved in conformity with the Record Description of the receiving item.

3. The three options, ZERO SUPPRESS, CHECK PROTECT, and FLOAT DOLLAR SIGN, all provide suppression of leading zeros and commas. If the LEAVING option is not employed, suppression will stop as soon as either a non-zero digit or the decimal point (actual or assumed) is encountered. Other pertinent factors are given below:

   a. When ZERO SUPPRESS is specified, leading zeros and commas will be replaced by spaces.

   b. When CHECK PROTECT is specified, leading zeros and commas will be replaced by asterisks.

   c. When FLOAT DOLLAR SIGN is specified, the rightmost character suppressed will be replaced by a dollar sign. All other characters which are suppressed will be replaced by spaces.

4. If the LEAVING option is used, the suppression or replacement will stop so as to leave the specified number of places to the left of the decimal point, unless it has stopped sooner by the above rule. The integer must be a numeric literal with a positive integral value.

5. When the BLANK WHEN ZERO clause is used, the item will be replaced by spaces if the value of the item is zero. Thus, all other editing requirements, such as ZERO SUPPRESS, CHECK PROTECT, etc., will be overridden.

6. When the PICTURE clauses and editing clauses appear in the same data description and contradict each other, the editing clauses are overridden.

7. More comprehensive editing features are available in the PICTURE clause. When any of the above clauses is used, the format of the item is assumed to contain editing symbols. Thus, the class of the item is alphanumeric.

LEVEL-NUMBER

Function

To show the hierarchy of data within a logical record. To identify entries for condition-names, independent constants, and independent working-storage items.

---

level-number

Notes:

1. A level-number is required as the first element in each Record Description entry.

2. A level-number may have values of 1-49, 77, or 88.

3. The level-number 1 signifies a logical record. This corresponds to the logical record on which the READ and WRITE verbs operate.

4. Special level-numbers have been assigned to certain entries where there is no real concept of level:

   a. Level-number 77 is assigned to identify independent constants, and independent working-storage items. Independent, in this case, means not part of any hierarchy.

   b. Level-number 88 is assigned to entries which define condition-names associated with a conditional variable. The conditional variable must be an elementary item.

OCCURS

Function

To describe a sequence of data items with the same format and to supply information required in the application of subscripts.

Option 1.

```
[, OCCURS integer-2 TIMES]
```

Option 2.

```
[, OCCURS [integer-1 TO] integer-2 TIMES DEPENDING ON
 data-name]
```

Notes:

1. Integer-1 and integer-2 must be numeric literals with positive integral values.

2. The OCCURS clause cannot be specified within the File-Section for an entry which has a level number of 01.

3. The OCCURS clause is used in defining tables and other homogeneous sets of repeated data. When the OCCURS clause is used, the data-name which is the subject of this entry must be subscripted whenever used as an operand in the Procedure Division. If this data-name is the name of a group item, then all data-names belonging to the group must be subscripted whenever used as operands.

The Record Description clauses associated with an item whose description includes an OCCURS clause apply to each repetition of the item being described.

4. This clause is required when either the data might not exist, or when it might occur more than once. If the clause is not used, the number of occurrences is assumed to be one.

5. In option 1, integer-2 represents the exact number of occurrences. It is illegal to have integer-2 equal to zero.

6. In option 2, integer-1 and integer-2 refer to the minimum and maximum number of occurrences, respectively. Integer-2 must always be greater than integer-1. If integer-1 is zero, the data may not be present; if it is 1, the data will be present, but need appear only once. Note that integer-1 does not imply that the item length is variable, but that the number of occurrences of the item may vary.

7. The use of option 2 means that the count of the occurrences of the data is equal to the value of the elementary item called data-name. This value must be a positive integer. If the data-name used in the DEPENDING option appears within the record in which the current Record Description entry also appears, then data-name must precede the variable portion of the record. In this case, integer-2 is considered as the maximum number of occurrences and is used for storage reservation.

8. Data-name in option 2 should be qualified when necessary, but subscripting is not permitted.

PICTURE

Function

To show a detailed picture of an elementary item, the general characteristics of the item, and special report editing.

[, PICTURE IS (any allowable combination of characters and symbols as described below)]

Notes:

Because the choice of characters in any given PICTURE depends on the type of data item being described, the characters will be grouped according to the type of data item they describe.

1. Numeric Items

   The PICTURE of any numeric data item may contain a combination of only the following characters:

   9 V P S

   The significance of these four characters is as follows:

   a. The character 9 indicates that the character position will always contain a numeric character.

   b. The symbol V indicates the position of an assumed decimal point. Since a numeric item cannot contain an actual decimal point, an assumed decimal point is used to provide the compiler with the information concerning the alignment of items involved in computation.

   c. The character P indicates an assumed decimal scaling position. It is used to specify the location of an assumed decimal point when the point is not within the data item. The character V may be used or omitted as desired. If it is used, however, it must be placed in the position of the assumed decimal point.

   d. The character S is equivalent to the SIGNED clause and indicates the presence of an operational sign. If used, it must always be written as the leftmost character of the PICTURE. If the USAGE of the item is COMPUTATIONAL, the character S in a PICTURE is a redundant specification. If the USAGE of the item is DISPLAY, it specifies a sign overpunch in the units position.

2. Alphabetic Items

The PICTURE of an alphabetic item can contain only the character A.

An A indicates that the character position will always contain an alphabetic character, i.e., a letter or a space.

3. Alphanumeric Items

An alphanumeric item has been defined as an item which may contain any character in the character set of the computer. Alphanumeric items can be divided into two types: non-report items for which editing is not specified, and report items for which editing has been specified.

a. Non-report Items.

The PICTURE of a non-report item may contain only the characters 9, A, and X and must contain at least one A or X. The characters 9 and A have been discussed above. A mixture is treated as if all the characters were Xs.

An X indicates that the character position may contain any character in the computer's character set.

b. Report Items.

Editing of numeric data is accomplished by moving the data to a report item which specifies the insertion, replacement, and/or suppression of certain characters. The PICTURE of a report item may contain a combination of the following characters:

9 V P , . + - Z * CR DB B 0 $

The uses of 9, V, and P have been discussed above. The remaining characters will be explained in three groups: zero suppression, insertion, and replacement characters.

Zero suppression or character replacement is accomplished by placing a character designated for the desired editing in each leading numeric character position that is to be suppressed or replaced. Two general rules apply:

(1) Suppression and/or replacement terminates with the character immediately preceding the first digit other than zero, or the decimal point (assumed or actual) whichever is encountered first.

(2) If all numeric character positions in a PICTURE reserved for source data (as opposed to those additional positions used for insertion characters) contain suppression characters (asterisk is excluded), then all characters will be replaced by spaces if the value of the source data is zero.

Zero Suppression Character. The character Z specifies that the character position is to be suppressed by a space if a non-significant zero appears in the position. Z must never be preceded by 9.

Insertion Characters. The single dollar sign character specifies that a $ is to be placed in the indicated position. The single dollar sign can be preceded only by the single plus or minus sign.

The single minus sign, written either as the first or the last character of a PICTURE, specifies that a display minus sign is to be placed in the indicated position when the value of the source item is negative. If the value is not negative, a space will be inserted.

The single plus sign, written either as the first or the last character of a PICTURE, specifies that a display minus sign is to be placed in the indicated position if the value of the source item is negative. If the value of the item is not negative, a display plus sign will be inserted.

The comma character specifies that a comma is to be inserted in the indicated position unless the following condition occurs: If the suppression or replacement has caused the elimination of all digits to the left of the comma, the comma itself will be suppressed.

The decimal point character specifies that an actual decimal point is to be inserted in the indicated position and the source item is to be aligned accordingly. Numeric character positions to the right of an actual decimal point in a PICTURE must consist of characters of one type.

CR and DB are called the "credit" and "debit" symbols and may appear only at the right end of a PICTURE. These symbols occupy two character positions and indicate that the specified symbol is to appear in the indicated positions if the value of a source item is negative. If the value is positive or zero, spaces will appear instead.

The zero specifies that the character 0 is to be inserted in the indicated position.

The character B specifies that a space is to be inserted in the indicated position.

Replacement Characters. The asterisk indicates check protection, i.e., the replacement of non-significant zeros by asterisks.

The Floating Dollar Sign. Zero suppression with a floating dollar sign is specified by placing a dollar sign in each numeric character position to be suppressed. A dollar sign will be placed in the rightmost position in which suppression is to occur.

The Floating Minus Sign. Zero suppression with a floating minus sign is specified by placing a minus sign in each numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur. If the value is positive or zero, a space will be inserted instead of a minus sign. All floating minus signs must be the leftmost characters in a PICTURE with the exception that commas may be imbedded. Suppression of leading commas is also provided.

The Floating Plus Sign. Zero suppression by means of a floating plus sign is specified by placing a plus sign in each leading numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur. If the value of the item is not negative, a plus sign will be inserted instead. All floating plus signs must be the leftmost characters in a PICTURE except that commas may be imbedded. Suppression of leading commas is also provided.

c. Scientific Decimal Items.

A scientific decimal item is a special type of report item which specifies editing of a floating point number. The PICTURE of a scientific decimal item may contain only the following characters:

+ - 9 . V E

Specifically the PICTURE must conform to the form:

$$\underbrace{\left\{\begin{matrix}+\\-\end{matrix}\right\} 9\ (m) \left[\begin{matrix}V\\.\end{matrix}\right] 9\ (n)\ E}_{\text{mantissa}} \quad \underbrace{\left\{\begin{matrix}+\\-\end{matrix}\right\} 99}_{\text{exponent}}$$

where m and n are positive integers and (m+n) must be less than 17. The symbol E is used to give visual separation of the exponent from the mantissa.

4. Record Mark.

The PICTURE character J specifies that the single character called record mark (of special importance to some peripheral equipment) is to appear in the indicated position as a constant. If J is used, the PICTURE must contain only one J, and no other character may appear with it.

The data item being specified is one alphanumeric character whose value is a record mark (≑) and all the rules for a data item with such specifications apply.

5. General Notes.

a. A PICTURE clause can be used only at the elementary item level.

b. An integer which is enclosed in parentheses following any symbol listed below indicates the number of consecutive occurrences of that symbol:

A X 9 P Z * $ B 0 - +

c. The maximum number of characters allowed in a PICTURE is 30.

d. All characters, other than the operational symbols (V, P, and S), are counted in the size of the item. Note that the CR and DB symbols occupy two character positions.

e. An item can possess only one sign, operational or display, and one decimal point, assumed or actual.

f. The maximum number of numeric character positions allowed in an item is 18, except in case of a scientific decimal item.

6. Summary

The chart below defines the allowable types of elementary items and gives the interrelationship of PICTURE, CLASS, and USAGE.

Type of Item	Class	USAGE	Characteristic Clause or Symbol in PICTURE	Legitimate Symbols in PICTURE
Alphanumeric (non-report)	AN or ALPHABETIC	DISPLAY	A or X	A X 9
Report	AN	DISPLAY	, . + - Z * CR DB B 0 or $ in PICTURE, BLANK WHEN ZERO, CHECK PROTECT, ZERO SUPPRESS, or FLOAT DOLLAR SIGN	9 V P , . + - Z CR DB B 0 $ *
External Decimal	NUMERIC	DISPLAY		9 V P S
Internal Decimal	NUMERIC	COMPUTATIONAL		9 V P S
Floating Point	NUMERIC	COMPUTATIONAL-1 COMPUTATIONAL-2		
Scientific Decimal	NUMERIC	DISPLAY	$\left\{\begin{matrix}+\\-\end{matrix}\right\}$ 9(m) $\left[\begin{matrix}V\\.\end{matrix}\right]$ 9(n) E $\left\{\begin{matrix}+\\-\end{matrix}\right\}$ 99	+ - 9 . V E

POINT LOCATION

Function.

To define an assumed decimal point.

$$\left[\text{, POINT LOCATION IS } \left\{ \begin{matrix} \text{LEFT} \\ \text{RIGHT} \end{matrix} \right\} \text{ integer } \left[\text{PLACE } \left[\text{S} \right] \right] \right]$$

Notes:

1. When the PICTURE clause is not given, the definition of numeric items having non-integral values requires the POINT LOCATION clause.

2. An actual decimal point may not be defined through the use of this clause. Actual decimal points must be shown in a PICTURE clause.

3. Integer must be a numeric literal with a positive value. The decimal point is located "integer" positions to the left or right of the least significant position of the field.

4. This clause can be used only at the elementary item level.

REDEFINES

Function.

To allow the same computer storage area to contain different data items, or to provide an alternative grouping or description of the same data.

```
level-number   data-name-1   [ REDEFINES   data-name-2 ]
```

Notes:

1. The REDEFINES clause, when used, must immediately follow data-name-1.

2. The level-numbers of data-name-1 and data-name-2 must be identical.

3. Redefinition starts at data-name-2 and ends when a level-number less than or equal to that of data-name-2 is encountered.

4. The storage area for data-name-1 must not be larger than that for data-name-2.

5. The entries giving the new description of the storage must immediately follow the entries describing the area being redefined.

6. This clause must not be used for logical records associated with the same file. The DATA RECORDS clause in the File Description implies automatic redefinition.

7. Data-name-2 should be qualified when necessary, but subscripting is not permitted.

8. The entries giving the new description of the storage area must not contain any VALUE clauses, except in condition-name entries.

SIGNED

Function.

To specify an operational sign for an elementary item.

```
[, SIGNED]
```

Notes:

1. An item which contains an operational sign must be numeric; therefore, its class need not be specified.

2. SIGNED indicates the use of a standard operational sign. (This may also be indicated by the use of an S in the PICTURE.) The standard operational sign is a sign overpunch in the low order position of a DISPLAY item and in the leftmost bit for a COMPUTATIONAL item. A COMPUTATIONAL item always has a sign; therefore, it is redundant to specify an operational sign. A standard operational sign is not considered in determining the size of an item.

3. When a SIGNED DISPLAY item is used as a source in a MOVE clause it may or may not actually have a sign overpunch. Sign overpunch is developed for a receiving item for negative values only.

4. An item which contains any editing symbols cannot have an operational sign.

5. This clause can be used only at the elementary item level.

SIZE

Function.

To specify the size of an item in terms of the number of characters or digits.

```
[, SIZE IS  integer  { CHARACTER [S] }  ]
                     { DIGIT [S]     }
```

Notes:

1. The size of an item must be specified at the elementary item level by means of a SIZE clause or a PICTURE. A PICTURE and a SIZE clause need not both be given. If both are given, they must agree. Use of the SIZE clause at any level other than the elementary item level is optional. The actual size of a group is the sum of the sizes of the elementary items comprising the group.

2. Any of the key words of the USAGE and/or CLASS clauses can be inserted between "integer" and the word CHARACTERS (or DIGITS) in the SIZE clause. If this is done, separate USAGE and/or CLASS clauses must not be written. An example of this is:

```
02  PAGE SIZE IS 7 NUMERIC COMPUTATIONAL DIGITS
    VALUE IS 0000342.
```

SYNCHRONIZED

Function.

To specify positioning of an elementary item within a computer word or words.

```
[, SYNCHRONIZED  { LEFT  }  ]
                 { RIGHT }
```

Notes:

1. When the SYNCHRONIZED clause is used the data item so described will appear in the immediately subsequent computer word(s).

2. For DISPLAY data items, SYNCHRONIZED LEFT specifies that the leftmost character of the item is to occupy the leftmost character position of a computer word. SYNCHRONIZED RIGHT specifies that the rightmost character of the item is to occupy the rightmost character position of a word.

3. A NUMERIC COMPUTATIONAL item specified as SYNCHRONIZED RIGHT will appear in the least significant position of a word (two words if more than ten digits) with its sign bit in the sign bit of the word. This is the most efficient form for use in computation.

4. If SYNCHRONIZED LEFT is specified, the leftmost character of the described item occupies the left-hand position of the next available machine word. This may mean that the right-hand portion of the preceding word is unoccupied. If the next described item is not SYNCHRONIZED (LEFT or RIGHT), the leftmost character of this item follows the rightmost character of the previous SYNCHRONIZED LEFT item.

5. A NUMERIC COMPUTATIONAL item specified as SYNCHRONIZED LEFT will occupy the least multiple of six bits capable of holding the item including its sign. The item will have its high order character (with its sign in the leftmost bit) in the leftmost portion of a computer word.

A NUMERIC COMPUTATIONAL item without a SYNCHRONIZED clause will also occupy the minimum number of six bit blocks capable of holding it and its sign (again in the leftmost bit). The item itself, however, may appear anywhere within a computer word, depending on where the previous item ended.

In both of the cases above, although the items are computational and therefore specified as being in binary form, they will occupy storage in six bit bytes.

USAGE

Function

To specify representation of a data item.

```
[, USAGE IS { COMPUTATIONAL   }  ]
            { COMPUTATIONAL-1 }
            { COMPUTATIONAL-2 }
            { DISPLAY         }
```

Notes:

1. The USAGE clause can be written at any level. If the USAGE clause is written at a group level, it applies to each elementary item in the group. The usage of an elementary item cannot contradict the usage of a group to which the item belongs.

2. The USAGE clause provides a description of the representation of data in storage. It will not convert data from one representation to another except in conjunction with Procedure Division clauses.

3. DISPLAY specifies storing of the item in BCD (character) form.

4. COMPUTATIONAL specifies storing of the item in binary form. A COMPUTATIONAL item is capable of representing a value to be used in computations and must be numeric. Therefore, the class need not be given for an item described as COMPUTATIONAL. If a group item is described as COMPUTATIONAL, the elementary items in the group are COMPUTATIONAL. The group item is not COMPUTATIONAL and cannot be used in computations. If the usage specified at the group level is suffixed, (-1 or -2), the elementary items are considered to be the same, and may not have a contradicting form of COMPUTATIONAL usage specified.

5. The three variations of COMPUTATIONAL indicate the following types of USAGE:

COMPUTATIONAL	internal decimal
COMPUTATIONAL-1	single precision floating point
COMPUTATIONAL-2	double precision floating point

The only clause required for a floating point item is USAGE.

6. If USAGE is not specified for an elementary item, or for any group to which the item belongs, the USAGE is assumed to be DISPLAY.

VALUE

Function.

To define the value of constants, the initial value of working-storage items or the values associated with a condition-name.

[, VALUE IS literal]

Notes:

1. The constant, initial, or conditional value produced by the use of a VALUE clause is the same as that which would result if the specified literal were placed in the item by means of the MOVE verb.

2. When the VALUE clause is used in a condition-name entry, no further information is required.

3. In the File Section, the VALUE clause can be used only in a condition-name entry.

4. In the Working-Storage Section, the VALUE clause is used to specify the initial value of an item and to specify the value associated with a condition-name.

5. In the Constant Section, the VALUE clause can be used only to specify a constant.

6. A figurative constant may be substituted for a literal in the format above.

7. The VALUE clause must not be stated in a Record Description entry which contains an OCCURS clause, or in an entry which is subordinate to an entry containing an OCCURS clause.

8. The VALUE clause cannot be used to specify the initial value of an item following a variable portion of a record defined by the DEPENDING option of an OCCURS clause.

9. When VALUE is not specified, no assumption should be made regarding the initial contents of the working-storage area.

10. All literals other than floating point literals have been discussed previously.

11. An initial value may be specified for a floating point or scientific Decimal item by means of a floating point literal. This genus of literal may appear only in a VALUE clause and must conform to the following rules:

    a. The plus or minus sign appears in the high order position (if a sign is given) and is the mantissa's sign. Absence of a sign implies a positive mantissa.

    b. The mantissa consists of from 1 to 16 digits and an optional real point. If a decimal point is specified, it may not appear to the right of the mantissa.

    c. Immediately to the right of the mantissa the exponent (if any is given) is represented by the symbol E, followed by the plus or minus sign (if a sign is given), and one or two digits. The magnitude of the exponent should not exceed 38. The value of the literal is the value of the mantissa multiplied by 10 raised to the power given by the exponent.

    d. Examples:

    The value 1 may be represented as

```
1
1000E-03
+.01E2
010.000E-1
```

12. If an external decimal item does not have a mathematical sign provision indicated (either by an S in its PICTURE clause or by a SIGNED clause). the sign of a negative value is lost.

Specific Entry for a Condition-Name

Each condition-name requires a separate entry with the level number 88. This entry contains the name of the condition and the value associated with the condition-name. The condition-name entries for a particular conditional variable must follow the entry describing the item with which the condition-name is associated.

SUMMARY

File-Section

Organization

The File Section contains a section-header, File Description entries, and Record Description entries.

Working-Storage Section

Concept of Working Storage

Working Storage is that part of storage set aside for intermediate processing of data. The difference between Working Storage and File Storage is that the former deals with the requirements for the storage of intermediate data results,whereas the latter deals with the charactistics of the entire file, as well as the requirements for the storage of each record of the file.

Organization

Whereas the File Section is composed of File Description entries and Record Description entries, the Working-Storage Section is composed only of Record Description entries. The Working-Storage Section begins with a section-header and a period, followed by Record Description entries for independent Working-Storage items, and then by Record Description entries for Working-Storage records, in that order.

Independent Working Storage

Items in Working Storage which bear no relationship to each other need not be grouped into records, provided they do not need to be further subdivided. Instead they are classified and defined as independent items. Each of these items is defined in a separate Record Description entry which begins with the special level number 77.

The following Record Description clauses are required in each entry:

a. level-number
b. data-name
c. CLASS or PICTURE
d. SIZE (when PICTURE is not specified)

The OCCURS clause is not meaningful and will cause an error at compilation time if used. Other Record Description clauses are optional and can be used to complete the description of the item, if necessary.

Working-Storage Records

Data elements in Working Storage which bear a definite relationship to each other must be grouped into records according to the rules for formation of Record Descriptions. All clauses which are used in normal input or output Record Descriptions can be used in a Working-Storage Record Description, including REDEFINES, and OCCURS. Each Working-Storage record-name (01 level) must be unique since it cannot be qualified by a file-name or section-name. Subordinate data-names need not be unique if they can be made unique by qualification.

Initial Values

The initial value of any elementary item in the Working-Storage Section may be specified by using the VALUE clause of the Record Description. All the rules for the expression of literals and figurative constants apply. Rules for a moving of the literal to the Working-Storage item also apply.

For example:

1. 77 PAGE SIZE IS 7 NUMERIC COMPUTATIONAL CHARACTERS VALUE IS 0000342. (Legal)

2. 04 PAGE-NBR SIZE 4 CLASS NUMERIC VALUE IS 5. (Legal)

3. 02 ADDRESS SIZE 5 CLASS AN VALUE IS XT1245Z. (Illegal)

4. 03 GROUPING SIZE 9 CLASS NUMERIC VALUE IS ZEROS. (Legal)

In example 2, the class is numeric. Therefore, the value of the item will be right justified with zeros in the high order positions and will be stored as 0005.

Condition-Names

Any Working-Storage item may constitute a conditional variable with which one or more condition-names may be associated. Entries defining condition-names must immediately follow the item to which they relate. Both the conditional variable entry and the associated condition-name entries may contain VALUE clauses. The conditional variable must be an elementary item.

Constant Section

Concept of Constant Storage

The concept of literals and figuratives enables the user to specify the value of a constant by writing its actual value (or a figurative representation of that value). It is often desirable to name this value and then refer to it by its name. For example, 6% (.06) may be named as INTEREST-RATE and then referred to by its name (INTEREST-RATE) instead of its value (.06).

Constant Storage is, therefore, that part of storage which is set aside to save named constants for use in a given program.

Organization

The Constant Section is organized in exactly the same way as the Working-Storage Section. It begins with a section-header, which is followed by Data Description entries for the independent constants, followed by the Record Description entries for the hierarchic constant records and their subordinate entries. This sequence should be followed.

Independent Constant Storage

Constants which bear no relationship to each other need not be grouped into records provided they need not be further subdivided. Instead they are classified and defined as independent constants. Each of these constants is defined in a separate Record Description entry which begins with the special level number 77.

The following Record Description clauses are required in each entry:

a. level-number

b. data-name

c. CLASS or PICTURE

d. SIZE (when PICTURE is not given)

e. VALUE

The OCCURS and REDEFINES clauses are not meaningful for independent constants and will cause an error at compilation time if used. Other Record Description clauses are optional and can be used to complete the description of the constant when necessary.

Constant Records

Named constants in the Constant Section which bear a definite relationship to one another must be grouped into records according to the rules for formation of Record Descriptions. All Record Description clauses can be used in a Constant Section Record Description. Each Constant Section record-name (01 level) must be unique since it cannot be qualified by a file-name or section-name. Subordinate data-names need not be unique if they can be made unique by qualification.

Value of Constants

In the definition of constants, the VALUE of every item must be specified individually. All the rules for the expression and movements of literals and figurative constants apply throughout this section.

Condition-Names

Since a constant can, by definition, have only one value, there can be no associated condition-names. The use of a condition-name entry (level 88) in the Constant Section is therefore illegal and will constitute an error in the source program.

Tables of Constants

Tables of constants, to be referred to by means of subscripting, are defined in the following way:

The table is defined as a record by a set of Record Description entries each of which specifies the VALUE of an element, or part of an element, of the table. In defining the record and its elements, the Record Description clauses(SIZE, USAGE, PICTURE, editing information, etc.) may be used to complete the definition where required. This form is required when the elements of the table require separate handling due to synchronization.
The hierarchical structure of the table is then shown by use of the REDEFINES entry and its associated subordinate entries. The subordinate entries, following the REDEFINES entry, must not contain VALUE clauses.

Part V: PROCEDURE DIVISION

The Procedure Division of a source program is used to specify the logical decisions and the actions which are to become the major portion of the compiled program.

GENERAL DESCRIPTION

The Procedure Division contains those procedures needed to solve a given problem. These procedures are written as sentences, combined to form paragraphs, which in turn may be combined to form sections.

RULES OF PROCEDURE FORMATION

General

COBOL procedures are expressed in a manner similar (but not identical) to common English prose. The basic unit of procedure formation is a sentence or a group of successive statements. A procedure is a paragraph, or a group of successive paragraphs, or a section, or a group of successive sections within the Procedure Division.

Statements

There are three types of statements: imperative statements, conditional statements, and compiler directing statements.

Imperative Statements

An imperative statement consists of a COBOL verb (excluding compiler directing verbs) and its operands.

Conditional Statements

A conditional statement is defined to be of one of the two following forms:

1. IF condition { statement-1 / NEXT SENTENCE } { OTHERWISE / ELSE } { statement-2 / NEXT SENTENCE }

2. Imperative statement followed by a related conditional statement.

In the second form, the imperative statement must not end with a GO TO or a STOP RUN statement. The ON SIZE ERROR option used with the arithmetic verbs, and AT END as used with the READ verb are the two allowable conditional statements of the second form.

Statement-1 or statement-2 can be either imperative or conditional and if conditional, can contain conditional statements in arbitrary depth. If statement-1 or statement-2 is conditional, then the conditions within the conditional statement are considered to be "nested." The phrase, OTHERWISE NEXT SENTENCE, may be omitted if it immediately precedes the period for the sentence. The same rule may be applied to the resulting sentence.

Compiler Directing Statements

A compiler directing statement consists of a compiler directing verb and its operands.

Sentence Punctuation

Verb Formats

See individual verbs for details.

Sentence Formats

The following rules apply to the punctuation of sentences:

1. A sentence is terminated by a period.
2. A separator is a word or character used for the purpose of enhancing readability. Use of a separator is optional.
3. The allowable separators are:
   ,
   THEN
4. These separators must not be followed immediately by another such separator.
5. Separators may be used in the following places:
   a. Between statements.
   b. In a conditional statement as defined under "Conditional Statements" (first form):
      (1) Between the condition and statement-1.
      (2) Between statement-1 and OTHERWISE.

Paragraphs

So that the source programmer may group several sentences to convey one idea (procedure), paragraphs have been included in COBOL. In writing procedures in accordance with the rules of the Procedure Division and the requirements of the Reference Format, the source programmer begins a paragraph with a name. A paragraph is terminated by the next paragraph or section name or by end of the Procedure Division.

Sections

A section consists of one or more successive paragraphs and when designated must be named. The section-name is followed by the word SECTION and a period. The section-name applies to all paragraphs following it until another section-name is found. It is not required that a program be broken into sections. For further details see Reference Format.

CONDITIONALS

General Description

Conditional procedures are extremely valuable in describing data processing problems. COBOL makes available to the programmer several means of expressing conditional situations.

COBOL conditionals generally involve the key word IF followed by the conditions to be examined followed by the operations to be performed. Depending upon the truth or falsity of the conditions different sets of operations are to be performed.

Conditions

Simple Conditions

A simple condition is one of four types of tests. These tests and the acceptable formats for stating them are described below.

1. Relation Tests. A relation test involves a comparison of two operands. Either of these two operands can be a data-name, a literal, or a formula. The comparison of two literals is not permitted. (Throughout the remainder of the discussion, the word "item" means either a data item or a literal.) Comparison of other numeric items is always permitted.

   a. Comparison of Numeric Items. For numeric items, a comparison results in the determination that the value of one of the items is less than, equal to, or greater than the other.

      The comparison of numeric items is based on the respective values of the items considered purely as algebraic values. The item length in terms of the number of digits is not itself significant. Zero is considered to represent a unique value regardless of the length, sign or implied decimal point location of an item.

b. Comparison of Non-numeric Items. For two non-numeric items, a comparison results in the determination that one of the items is less than, equal to, or greater than the other with respect to a collating sequence. The standard collating sequence is the 7090/94 collating sequence, but an alternate sequence may be specified by designating that the commercial collating sequence is to be used for non-numeric comparisons. This is specified in the Environment Division OBJECT-COMPUTER paragraph.

The figurative constant HIGH-VALUE [S] assumes the value of left parenthesis in the 7090/94 collating sequence and 9 in the commercial collating sequence. The figurative constant LOW-VALUES [S] assumes the value of zero in the 7090/94 collating sequence and blank in the commercial collating sequence.

COLLATING SEQUENCE

The two permissible collating sequences, in order, with the lowest values at the top of the columns, are as follows:

7090/94	commercial
0 through 9	blank or space
=	.
'	) or ¤
+	+ or &
A through I	$
¢	*
.	-
)	/
-	,
J through R	(or %
¬	= or #
$	' or @
*	¢
blank or space	A through I
/	¬
S through Z	J through R
≠	≠
,	S through Z
(	0 through 9

2. Full Relation Test Format. There are two cases to consider: equal length items, and unequal length items.

a. Items of Equal Length. If the items are of equal length, comparison proceeds by comparing characters in corresponding character positions starting from the high order end and continuing until either a pair of unequal characters is encountered or the low order end of the item is reached, whichever comes first. The items are determined to be equal when the low order end is reached.

The first encountered pair of unequal characters is compared for relative location in the ordered character set. The item which contains that character which is positioned higher in the ordered sequence is determined to be the greater item.

b. Items of Unequal Length. If the items are of unequal length, comparison proceeds as described above. If this process exhausts the characters of the shorter item, then the shorter item is less than the longer item unless the remainder of the longer item consists solely of spaces, in which case the two items are equal.

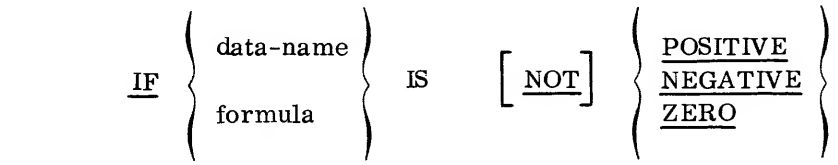
The format for full relation tests is:

```
                 ⎧ data-name-1 ⎫   ⎧ IS [NOT] GREATER THAN ⎫   ⎧ data-name-2 ⎫
                 ⎪             ⎪   ⎪ IS [NOT] LESS THAN    ⎪   ⎪             ⎪
IF               ⎨ literal-1   ⎬   ⎨ IS [NOT] EQUAL TO     ⎬   ⎨ literal-2   ⎬
                 ⎪             ⎪   ⎪ IS [NOT] =            ⎪   ⎪             ⎪
                 ⎩ formula-1   ⎭   ⎪ IS UNEQUAL TO         ⎪   ⎩ formula-2   ⎭
                                   ⎪ EQUALS                  ⎪
                                   ⎩ EXCEEDS                 ⎭
```

The word EXCEEDS is equivalent to IS GREATER THAN. The phrase IS UNEQUAL TO is equivalent to IS NOT EQUAL TO.

In the above format, the actual choice from data-name-1, literal-1, or formula-1 is called the subject. The choice from data-name-2, literal-2, or formula-2 is called the object. The subject and the object cannot both be literals.

An alternative way of stating a comparison of the value zero with a formula, or with an item whose class is numeric, is provided by the following form:



The specific interpretations of these terms are as follows: an item or formula is positive only if its value is greater than zero; an item or formula whose value is zero is not positive; an item or formula is negative only if its value is less than zero; an item or formula whose value is zero is not negative. In brief, the value zero is considered neither positive nor negative.

3. Conditional Variable Test. An item whose specific values can be named is called a conditional variable. A name given to a set of one or more of these values is called a condition-name.

   A conditional variable test is one in which an item is tested to determine whether or not one of the values associated with a condition-name is present.

   The format for a conditional variable test is:

   IF condition-name

4. Switch Status Test. The format for a switch status test is:

   IF condition-name

   In the switch status test the condition-name is associated with the ON or OFF status of a switch. The switch must be named in the SPECIAL-NAMES paragraph of the Environment Division.

Compound Conditions

Simple conditions can be combined with logical operators according to specified rules to form compound conditions. The logical operators are AND, OR, and NOT, and are defined by the following table, where A and B represent simple conditions:

A	B	NOT A	A AND B	A OR B
True	True	False	True	True
False	True	True	False	True
True	False	False	False	True
False	False	True	False	False

Thus, if A is true and B is false, then the expression A AND B is false, while the expression A OR B is true.

The rules for determining the truth value of a compound condition are as follows:

1. If AND is the only logical connective used, then the compound condition is true only if each of the simple conditions is true.

2. If OR is the only logical connective used, then the compound condition is true only if one or more of the simple conditions is true.

3. If both logical connectives are used (if both AND and OR appear), then there are two cases to consider, depending on whether or not parentheses are used.

   a. Parentheses can be used to indicate grouping. They must always be paired, as in algebra, and the expressions within the parentheses will be evaluated first. The precedence of nested parenthetical expressions is the same as normal algebra. That is, the innermost parenthetical expressions are evaluated first.

   b. If parentheses are not used, then the conditions are grouped first according to AND, proceeding from left to right, and then by OR, proceeding from left to right.

Examples:

(1) To evaluate C1 AND (C2 OR NOT (C3 OR C4)), use the first part of rule 3 and successively reduce this by substituting as follows:

Let C5 equal C3 OR C4 resulting in C1 AND (C2 OR NOT C5)

Let C6 equal C2 OR NOT C5 resulting in C1 AND C6

This can be evaluated by the table shown above.

(2) To evaluate C1 OR C2 AND C3, use the second part of rule 3 and reduce this to C1 OR (C2 AND C3), which can now be reduced as in example (1).

(3) To evaluate C1 AND C2 OR NOT C3 AND C4, group first by AND from left to right, resulting in:

(C1 AND C2) OR (NOT C3 AND C4)

which can now be reduced as in example (1).

(4) To evaluate C1 AND C2 AND C3 OR C4 OR C5 AND C6 AND C7 OR C8, group from the left by AND to produce:

((C1 AND C2) AND C3) OR C4 OR ((C5 AND C6) AND C7) OR C8

which can now be reduced as in example (1).

Table of Legal Symbol Pairs Involving Conditions and Logical Connectives

		SECOND SYMBOL					
		C	OR	AND	NOT	(	)
	C	-	P	P	-	-	P
FIRST SYMBOL	OR	P	-	-	P	P	-
	AND	P	-	-	P	P	-
	NOT	P*	-	-	-	P	-
	(	P	-	-	P	P	-
	)	-	P	P	-	-	P

Where P indicates that the pair is permissible, and the dash indicates a symbol pair that is not permissible, the pair OR NOT is permissible, while the pair NOT OR is not permissible.

* Permissible only if the condition itself does not contain a NOT.

FORMULAS

General

A formula is an algebraic expression consisting of a combination of arithmetic operators and data-names and/or literals, representing items on which arithmetic may be performed.

Basic Operators

There are five arithmetic operators which may be used in formulas. They are expressed by characters.

Operation	Character
(Addition)	+
(Subtraction)	-
(Multiplication)	*
(Division)	/
(Exponentiation)	**

The rules for forming algebraic expressions assume the existence of a precedence table for the arithmetic operators which determines the sequence in which the arithmetic operations in a formula will be executed unless parenthesizing is used to modify the hierarchy.

Normal precedence, from high to low, is:

Exponentiation
Multiplication and Division
Addition and Subtraction

Parentheses may be said to have a precedence higher than any of the operators, and are used to eliminate ambiguities in logic where consecutive operations of the same hierarchical level appear, or to modify the normal hierarchical sequence of execution in formulas where it is necessary to have some deviation from the normal precedence. When the sequence of execution is not specified by parentheses, the order of execution of consecutive operations of the same hierarchical level is from left to right. Thus, expressions ordinarily considered to be ambiguous, e.g., A/B*C and A/B/C are permitted in COBOL. They are interpreted as if they were written (A/B) * C, and (A/B) / C, respectively. Without parenthesizing, the following example illustrates normal precedence:

```
A + B / C + D ** E * F - G
```

would be interpreted as if written:

```
A + (B / C) + ((D ** E) * F) - G
```

with the sequence of operations working from the inner-most parentheses toward the outside. Exponentiation will be performed first, then multiplication and division and finally addition and subtraction. Exponentiation of a negative variable or literal is allowed only if the exponent is a literal equal to zero, one, or two.

Formation Of Symbol Pairs

The ways in which symbol pairs may be formed are summarized in the table below, where P indicates a permissible pair.

		SECOND SYMBOL				
		VARIABLE	* / **	-+	(	)
FIRST SYMBOL	VARIABLE	-	P	P	-	P
	* / **	P	-	P	P	-
	+ -	P	-	-	P	-
	(	P	-	P	P	-
	)	-	P	P	-	P

VERBS

Listed By Categories

Arithmetic	ADD SUBTRACT MULTIPLY DIVIDE COMPUTE
Input-Output	READ WRITE OPEN CLOSE DISPLAY
Procedure Control	GO TO ALTER PERFORM
Data Movement	MOVE
Ending	STOP
Compiler Directing Verbs	ENTER EXIT NOTE

Note: Although the word IF is not a verb in the strictest sense, it possesses one of the most important characteristics of one, namely the generation of coding in the object program. Its occurrence is a vital feature in the Procedure Division.

Specific Verb Formats

The specific verb formats, together with a detailed discussion of the restrictions and limitations associated with each, appear on the following pages, in alphabetic sequence.

ADD

Function

To add two or more numeric data items and set the value of an item equal to the result.

```
ADD  { literal-1 }  [ ,  { literal-2 }  ... ]  [ { TO     }  data-name-n ]
     { data-name-1 }     { data-name-2 }        { GIVING }

     [ ROUNDED ]  [ , ON SIZE ERROR  { imperative-statement-1 }
                                     { NEXT SENTENCE          } ]

     [ { ELSE      }  { imperative-statement-2 } ]
     [ { OTHERWISE }  { NEXT SENTENCE          } ]
```

Notes:

1. The data-names used must refer only to the special register, TALLY, or to numeric elementary items whose descriptions appear in the Data Division of the program.

2. The maximum size of any operand (literal or data-name) is 18 decimal digits. An error will be indicated at compilation time if the format for any operand specifies a number of digits in excess of 18. Intermediate results are carried to a maximum of 20 digits with no loss of least significant digits except when the maximum is reached.

3. If the GIVING option is used, the value of data-name-n will be made equal to the sum of the values of the preceding data-names and/or literals. Data-name-n is not used as an addend in this option; hence its format may contain editing symbols.

If the TO option is used, the sum of the values of data-name-n and the preceding data-names and/or literals will be calculated; the value of data-name-n will then be made equal to the sum.

If neither the GIVING nor the TO options are used, the last named (the rightmost or last written) addend must not be a literal.

The sum of the values of all the data-names and/or literals will be calculated. The value of the rightmost data-name will then be made equal to the sum. Since it is used as an addend in this case the format of the rightmost data-name may not contain any editing symbols.

Examples:

Statement	Sum	Stored In
ADD A, B, C	A+B+C	C
ADD A, B, TO C	A+B+C	C
ADD A, B, C GIVING D	A+B+C	D

4. An error will be indicated at compilation time if the data description of any item used as an addend specifies the presence of editing symbols. Operational signs and implied decimal points are not considered editing symbols. Literals used as addends must be numeric.

5. An ADD statement must refer to at least two addends.

6. The formats associated with all operands referred to in an ADD statement may differ among each other. Decimal point alignment is automatically supplied throughout the calculation.

7. If the number of decimal places in the calculated result (sum) is greater than the number of decimal places associated with the resultant data-name (the data-name whose value is to be set equal to the sum), truncation will occur unless the ROUNDED option has been specified. Truncation is always in accordance with the size associated with the resultant data-name. When the ROUNDED option is specified, the least significant digit of the resultant data-name has its value increased by 1 whenever the most significant digit of the excess is greater than or equal to 5.

8. Whenever the number of integral places (those to the left of the decimal point) in the calculated result exceeds the number of integral places associated with the resultant data-name, a size error condition arises.

In the event of a size error condition, one of two possibilities will occur, depending on whether or not the ON SIZE ERROR option has been specified.

a. The testing for the size error condition occurs only when the ON SIZE ERROR option is specified in the verb format. In the event that ON SIZE ERROR is not specified, and a size error condition arises, the effect will be unpredictable.

b. If the ON SIZE ERROR option has been specified, and a size error condition arises, the value of the resultant data-name will be altered unpredictably, and the imperative-statement-1 associated with the ON SIZE ERROR option will be executed.

ALTER

Function

To modify a predetermined sequence of operations.

```
ALTER   procedure-name-1      TO PROCEED TO  procedure-name-2
      [ procedure-name-3      TO PROCEED TO  procedure-name-4 ... ]
```

Notes:

1. Procedure-name-1, procedure-name-3, ..., are names of paragraphs, each containing a single sentence consisting of only a GO TO statement as defined under option 1 of the GO TO verb.

2. The effect of an ALTER statement is to replace the procedure-name specified in the GO TO sentence (located at procedure-name-1) by the procedure-name-2 specified in the ALTER statement.

CLOSE

Function

To terminate the processing of input and output reels and files, with optional rewind and/or lock.

```
CLOSE   file-name-1  [ REEL ]  [ WITH { NO REWIND } ] [ , file-name-2 ... ]
                                      { LOCK      }
```

Notes:

1. The CLOSE file-name option (as applied to the entire file rather than to individual reels) will initiate the final closing conventions for the file and release the data area. The CLOSE verb may be applied to any file with OPEN status but must not be re-used on the same file without an intervening OPEN instruction.

2. CLOSE file-name (without the REEL option) will have the following effects:

   a. Input Files.

      (1) If neither NO REWIND nor LOCK is specified, the current reel of the file will be rewound.

      (2) If the NO REWIND option is specified, the current reel of the file will not be rewound.

      (3) If the LOCK option is specified, the current reel of the file will be rewound and unloaded.

   Note that any label processing is performed only when the physical end-of-file is encountered on the tape. This condition normally occurs prior to the closing of the file.

   b. Output Files.
   The final closing conventions for the file are performed and the data area is released. Furthermore,

      (1) If neither LOCK nor NO REWIND is specified, the current reel of the file will be rewound.

      (2) If the NO REWIND option is used on a tape file, the last reel of the file will remain positioned at the end of the file.

      (3) The use of the LOCK option will rewind and unload the current reel.

3. If the CLOSE file-name REEL option is used, then, for both input and output files, the next reel processing procedures are instituted. More specifically:

   a. Input Files.

      Processing of the end label will be bypassed but the procedures for checking the label on the next reel will be executed. If a CLOSE REEL is given for the last reel of a file, an error will occur in the object program. Furthermore,

(1) If neither LOCK nor NO REWIND is specified, the current reel will be rewound.

(2) If the NO REWIND option is used, the current reel is not rewound. However, this may cause an error in the object program.

(3) If the LOCK option is used, the current reel will be rewound and unloaded.

(4) RERUN procedures are not executed for the file.

b. Output Files.

The standard end-of-reel processing takes place immediately. Furthermore,

(1) If neither LOCK nor NO REWIND is specified, the current reel will be rewound.

(2) If the NO REWIND option is used, the current reel is not rewound. However, this may cause an error in the object program.

(3) If the LOCK option is used, the current reel will be rewound and unloaded.

(4) RERUN procedures for the file, if specified, are not executed.

4. When a CLOSE REEL is given, the locking and rewinding options of the CLOSE REEL will take precedence only for the current reel and regardless of the options associated with a CLOSE of file. When a CLOSE file-name is given, its options will be executed for the current reel of the file.

5. If the file has been specified as OPTIONAL (see the FILE-CONTROL paragraph of the Environment Division), the standard end-of-file processing is not performed whenever this file is absent.

6. If a file is assigned to a system unit, the CLOSE options for the file may be overridden by the CLOSE options characteristic of the system unit. For instance, the options concerning the closing of the IBSYS unit SYSOU1 are under control of the compiler regardless of the CLOSE options specified in the source program.

COMPUTE

Function

To permit use of formulas.

```
COMPUTE data-name-1 [ ROUNDED ] { FROM   } { data-name-2 }
                                { =      } { formula   }
                                { EQUALS }

    [ , ON SIZE ERROR { imperative-statement-1 }
                      { NEXT SENTENCE          }

      [ { ELSE      } { imperative-statement-2 } ] ]
      { OTHERWISE } { NEXT SENTENCE          }
```

Notes:

1. Data-name-1 may not be a literal.

2. The data-name-2 option provides a method, other than MOVE, for making the value of data-name-1 equal to the value of data-name-2.

3. The formula option permits the use of any meaningful combination of numeric literals, arithmetic operators, and data-names. These may all be parenthesized as required. Any data-names used must satisfy the general rules specified for data-names used with the simple arithmetic verbs.

4. The ON SIZE ERROR option applies only to the final result and does not apply to any of the intermediate results.

5. All rules regarding the ON SIZE ERROR option, the ROUNDED option, the size of operands, truncation, and the editing of results (which are specified for the simple arithmetic verbs) apply also to the verb COMPUTE. (See ADD.)

6. The words FROM and EQUALS are equivalent to each other and to the equal sign symbol. They may be used interchangeably, and the choice is generally made for readability.

7. Data-name-1 or data-name-2 may be the special register, TALLY.

8. ADD, SUBTRACT, MULTIPLY, and DIVIDE are converted to the corresponding COMPUTE statements by the compiler.

DISPLAY

Function

To display low volume data on an available hardware device.

```
DISPLAY  { literal-1   }  [ , { literal-2   }  ... ]
         { data-name-1 }      { data-name-2 }
```

Notes:

1. The standard DISPLAY device is the on-line printer.

2. When DISPLAY is followed by multiple operands, the data comprising the first operand is displayed as the first set of characters, the data comprising the second operand as the second set of characters, and so on. An extra space is inserted after each set.

3. For any single data item or literal in 7090/7094 COBOL, the maximum number of characters which may be displayed is 72.

4. Internal decimal data items (USAGE COMPUTATIONAL) are prepared for external output with a sign overpunch indicated over the rightmost position when the items are minus. The printer recognizes this as an alphanumeric character and prints it accordingly. The user must interpret this character. Floating point data items are displayed in the scientific decimal form. Other data items are displayed as they appear in core storage.

5. If an output file is assigned to the same device that is used for DISPLAY statements, WRITE statements on that file and DISPLAY statements on that device will not necessarily appear on the listing in the order that the statements of the source program were written. Output resulting from a WRITE statement is buffered whereas output resulting from a DISPLAY statement is produced immediately.

DIVIDE

Function

To divide one numerical data item into another and set the value of an item equal to the result.

```
DIVIDE { data-name-1 } INTO { data-name-2 } [GIVING data-name-3]
       { literal-1   }      { literal-2   }

    [ROUNDED] [, ON SIZE ERROR { imperative-statement-1 }
                               { NEXT SENTENCE          }

    [{ELSE     } { imperative-statement-2 }]]
     {OTHERWISE} { NEXT SENTENCE          }
```

Notes:

1. The data-names used must refer to the special register, TALLY, or numeric elementary items whose descriptions appear in the Data Division of the program.

2. All rules specified under the verb ADD regarding the size of operands, the ON SIZE ERROR option, the ROUNDED option, the GIVING option, truncation, and the editing of results, apply to the DIVIDE verb. (See ADD.)

3. An error will be indicated at compilation time if the data description for either data-name-1 or data-name-2 specifies the presence of editing symbols. Literals used must be numeric.

4. When the GIVING option is not specified, a literal must not be used as the dividend.

5. Division by zero constitutes a special type of size error. Program control may be provided through the use of a test for zero prior to attempting division. If the zero test type of program control is not provided, the rules specified under the ADD verb with respect to the ON SIZE ERROR option apply.

ENTER

Function

To permit communication between a COBOL object program and one or more subroutines assembled by IBMAP.

ENTER LINKAGE-MODE.

CALL 'entry-name-1' [USING data-name-1 [, data-name-2...]]

[RETURNING procedure-name-1 [,procedure-name-2...]] .

ENTER COBOL.

Notes:

1. ENTER LINKAGE-MODE and ENTER COBOL do not themselves cause generation of instructions but do permit the use of the special non-COBOL verb CALL.

2. Each time LINKAGE-MODE is entered, CALL may be used as many times as desired, but normal procedure statements may not be used again until ENTER COBOL has been given.

3. A CALL statement may have a paragraph-name in the A-margin.

4. The normal rules of program flow apply to CALL statements. That is, a CALL statement may follow any main routine statement. A paragraph containing a CALL statement may also be used in a PERFORM statement.

5. Entry-name identifies the entry point of the subordinate program. If the subordinate program was separately assembled under IBMAP for inclusion with an IBCBC program at load time, the entry name will correspond to the name given in columns 8-13 of the $IBMAP card. Entry-name must be surrounded by quotes so that the compiler will treat it as a non-numeric literal. This allows for use of names permissible in IBMAP but not normally permissible in COBOL.

6. The USING clause provides a means of informing a subordinate program of the object-time locations of particular data items. Such data items should be described in the Working-Storage or Constant Sections since input/output records are variably located and processed in IOCS buffers.

7. Since the USING clause communicates to a subordinate program the machine word locations but not the initial character positions of data items, it is useful to specify such items as SYNCHRONIZED.

8. The RETURNING clause permits communication from the subordinate program to the COBOL program by means of multiple return points. Although return from a subordinate program is normally to the statement following the CALL statement, additional return points may be indicated by use of RETURNING and additional procedure-names. Note that the subordinate program may also communicate with the main program through alteration of the contents of data items referred to by the USING clause.
9. For information about the construction of the subordinate program, consult the IBMAP literature, giving particular attention to the CALL, RETURN, and SAVE macro-instructions.

EXIT

Function

To provide a paragraph-name which may serve as a reference end point for PERFORM.

---

paragraph-name EXIT.

Notes:

1. EXIT must be preceded by a paragraph-name and appear as a single, one-word paragraph whose content is non-existent.
2. When a series of procedures is under the control of a PERFORM verb and the logical flow demands an ultimate common transfer point, this paragraph provides the predicate for the various GO TO instructions.
3. When the PERFORM verb is used, an EXIT paragraph-name may be the procedure-name given as the object of the THROUGH option.
4. In all other cases EXIT paragraphs perform no function and sequential control passes through them to the first sentence of the next paragraph.

GO TO

Function

To depart from the normal sequence of procedures.

Option 1:

GO TO [procedure-name-1]

Option 2:

GO TO procedure-name-1, procedure-name-2

[procedure-name-3 . . .] DEPENDING ON data-name

Notes:

1. When using option 1, if the GO TO statement is to be modified by the ALTER verb:
   a. The GO TO statement must itself have a paragraph-name.
   b. The paragraph in which the GO TO statement is included must consist solely of the GO TO statement.

   The paragraph-name assigned to the GO TO statement is referred to by using ALTER verb in order to modify the sequence of the program. If procedure-name-1 is omitted, and if the GO TO statement is not referred to by an ALTER statement prior to the first execution of the GO TO statement, execution of the program will be terminated and control will be returned to IBJOB.

2. In option 2, the contents of data-name must have a positive integral value at object time. The branch will be to the 1st, 2nd, ..., nth procedure-name, as the value of data-name is 1, 2, ..., n. If the value of data-name is anything other than the integers 1, 2, ..., n, then no transfer is executed and control passes to the next statement in the normal sequence for execution.

MOVE

Function

To transfer data, in accordance with the rules of editing, to one or more data areas.

```
                  ⎧ data-name-1                                    ⎫
MOVE              ⎪ CORRESPONDING    data-name-1                   ⎪    TO    data-name-2
                  ⎩ literal                                        ⎭

                        [ , data-name-3 . . . ]
```

Notes:

1. Additional receiving areas may be given, following data-name-2. The data designated by the literal or data-name-1 will be moved first to data-name-2, then to data-name-3, etc. When data-name-2 is referred to in this discussion, the note also applies to the other receiving areas.

2. It is improper to use MOVE (without the CORRESPONDING option) for a group item whose format is such that editing would be required on the elementary items in separate operations. If this type of procedure is desired, the CORRESPONDING option must be used, or else each elementary item must be handled individually by means of the verb, MOVE.

3. At object time, data is stored in conformity with the description of the receiving area. When the sizes of the areas of two group items involved in a move are not the same, a warning will be given by the compiler during compilation. A warning is also given for a move from an elementary to a group item or vice versa.

4. When numeric elementary items are moved, they are subject to the following procedures:

   a. They are aligned by decimal points, with zero filling or truncation on either end as required. A warning is given by the compiler if significant digits will be lost through truncation.

   b. They may be converted from one form to another, e.g., internal decimal to external decimal, numeric mode to alphanumeric mode.

   c. They may have special editing performed on them with suppression of zeros, insertion of dollar sign, commas, a decimal point, etc., and alignment as specified by the description of the receiving area. The presence of these special characters in an item actually makes the item alphanumeric. If such an item is referred to a source, the special characters will be picked up as a part of the data, and the verb making the reference will treat this data according to the rules specified for the treatment of alphanumeric data.

5. When non-numeric items are moved:

   a. The characters are placed in the receiving area left to right.

   b. If the source field is shorter than the receiving field, the remaining character positions are filled with spaces.

   c. If the source field is longer than the receiving field, the generated code provides for termination of the move as soon as the receiving field has been filled. A warning message is given for this condition unless one has already been given as stated in note 3.

   d. A table of legal moves, using the verb MOVE, is given below.

   A detailed description of the types of fields represented may be found under the PICTURE clause in the Data Division. Numbers in parentheses in the table refer to subsequent notes.

Source Field Type	Receiving Field Type: Group Item (1)	Alphabetic	Alphanumeric (SIZE > 18)	Alphanumeric (SIZE ≦ 18)	Report	External Decimal	Internal Decimal	Floating Point	Scientific Decimal
Group Item (1)	yes	yes	yes	yes	no	no	no	no	no
Alphabetic	yes	yes	yes	yes	no	no	no	no	no
Alphanumeric (SIZE > 18)	yes	yes	yes	yes	no	no	no	no	no
Alphanumeric (SIZE ≦ 18)	yes	yes	yes	yes	yes(4)	yes(4)	yes(4)	yes(4)	yes(4)
Report	yes	no	yes	yes	no	no	no	no	no
External Decimal	yes	no	yes	yes(5)	yes	yes	yes	yes	yes
Internal Decimal	yes	no	yes	yes(5)	yes	yes	yes	yes	yes
Floating Point	yes	no	yes	yes(5)	yes	yes	yes	yes	yes
Scientific Decimal	yes	no	yes	yes(5)	yes	yes	yes	yes	yes
ZERO [S] or ZEROS	yes	no	yes	yes	yes(2)	yes	yes	yes	yes(2)
SPACE [S]	yes	yes	yes	yes	yes(3)	yes(3)	no	no	yes(3)
LOW-VALUE [S]	yes	no	yes	yes	no	no	no	no	no
HIGH-VALUE [S]	yes	no	yes	yes	no	no	no	no	no
QUOTE [S]	yes	no	yes	yes	no	no	no	no	no
ALL---	yes	no	yes	yes	no	yes	no	no	no

(1) Group items are treated as having a PICTURE of all X's.

(2) The value zero is moved in accordance with editing requirements.

(3) A warning message is given.

(4) The source field is treated as an integer with sign over the units position.

(5) The receiving field is considered to have a PICTURE of all 9's (with a sign over the units position when minus).

e. For source fields of the scientific decimal type:

At object time a free form of data is allowed within the limits of the field. For example, a field with PICTURE -99V9E-99 may contain the value 1 in any of the following ways (where b represents a space):

b . 01b2b	(01×10^2)
1bb+01b	(note scale applied when no point)
.001bb3	
b10bbbb	(note scale applied when no point)
1000.-3	
etc.	

Note that the letter E is never part of the data.

6. If the CORRESPONDING option is used, selected items within data-name-1 are moved, with any required editing, to selected areas within data-name-2. Items are selected by matching the data-names of areas defined within data-name-1 with like data-names of areas defined within data-name-2, according to the following rules:

   a. At least one of the items of a selected pair must be an elementary item.

   b. The respective data-names are the same including all qualification up to but not including data-name-1 and data-name-2.

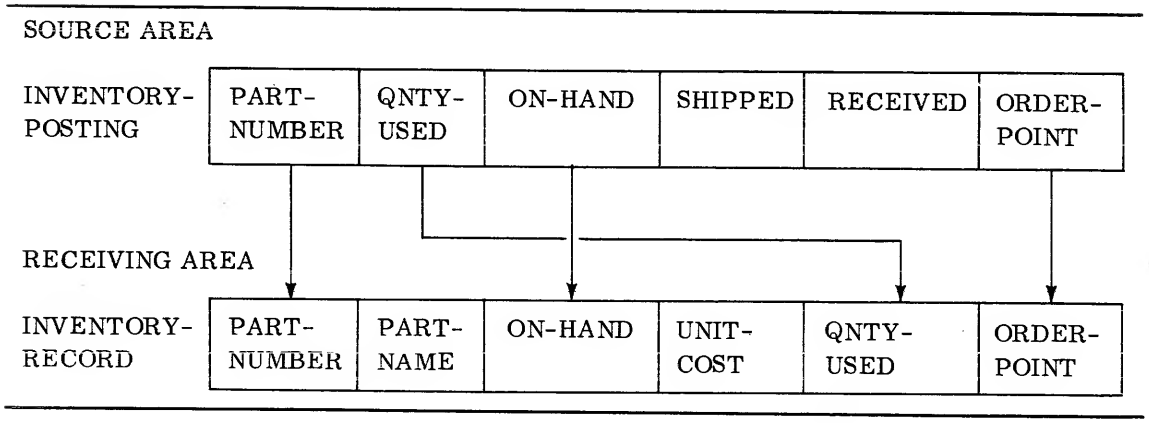
   Each CORRESPONDING source item is moved in conformity with the description of the receiving area. The results are the same as if the user had referred to each pair of CORRESPONDING data-names in separate MOVE statements.

7. When using a MOVE CORRESPONDING, only the first complete description of any area will be considered in the case where a REDEFINE has been used. All items describing the data contained within a REDEFINE group will be ignored.

8. If the CORRESPONDING option is used, no items in the group referred to can contain an OCCURS clause.

9. A MOVE CORRESPONDING must not refer to items having level numbers 77 or 88.

10. The following is an example of MOVE CORRESPONDING. Note the non-corresponding items in the source area are not moved and the non-corresponding items in the receiving area are not affected.



MULTIPLY

Function

To multiply two numeric data items and set the value of an item equal to the result.

```
MULTIPLY { data-name-1 } BY { data-name-2 } [ GIVING data-name-3 ]
         { literal-1   }    { literal-2   }

[ ROUNDED ] [ , ON SIZE ERROR { imperative statement-1 } ]
            {                 { NEXT SENTENCE      }   }

[ { ELSE      } { imperative statement-2 } ] ]
  { OTHERWISE }   { NEXT SENTENCE        }   ]
```

Notes:

1. The data-names used must refer to the special register, TALLY, or numeric elementary items whose descriptions appear in the Data Division.

2. All rules specified under the verb ADD with respect to the size of operands, the ON SIZE ERROR option, the ROUNDED option, the GIVING option, truncation, and the editing of results, also apply to the MULTIPLY verb. (See ADD.)

3. The formats of data-name-1 and data-name-2 may never contain editing symbols (dollar signs, commas, etc.). Implied decimal points and operational signs are not considered editing symbols.

4. A literal must not be specified as the multiplier unless the GIVING option is used.

NOTE

Function

To allow the programmer, in the Procedure Division of the source program, to write explanatory statements which will be produced on the listing but not compiled.

NOTE ...

Notes:

1. Following the word NOTE, any combination of the characters from the allowable character set may appear.

2. If NOTE is the first verb of a paragraph, the entire paragraph must be notes. Proper format rules for paragraph structure must be observed.

3. If NOTE is not the first verb of a paragraph, the commentary ends with a period followed by a space.

OPEN

Function

To initiate the processing of both input and output files. Performs checking or writing of standard labels, and other input/output functions.

```
                ⎧ [INPUT  file-name-1                    [, file-name-2 ...]] ⎫
OPEN            ⎨ [OUTPUT file-name-3                    [, file-name-4 ...]] ⎬
                ⎩ [INPUT  file-name-1  ...  OUTPUT  file-name-3 ...         ] ⎭
```

Notes:

1. At least one file must be named when the OPEN verb is used.

2. The verb OPEN must be executed prior to the first READ or WRITE for any file.

3. A second OPEN of a file cannot be executed prior to the execution of a CLOSE of the file.

4. The OPEN does not obtain or release the first data record. A READ or WRITE, respectively, must be executed to obtain or release the first data record.

5. OPEN initiates the standard label checking or writing procedures under the input/output system.

6. If an input file has been designated as optional in the FILE-CONTROL paragraph of the Environment Division, the object program will cause an interrogation as to the presence or absence of this file. If the reply to the interrogation is negative, i.e., the file is not present, the OPEN will not be executed, an indication of the absence of the file will occur, and an end-of-file signal will be sent to the Input/Output Control System of the object program. Thus, when the first READ for this file is encountered, the end-of-file path for this statement will be taken. The interrogation for the presence or absence of a file is accomplished by determining if there is a unit assigned to the file (see FILE-CONTROL paragraph, ASSIGN TO NONE clause).

PERFORM

Function

To depart from the normal sequence of procedures in order to execute one statement, or a sequence of statements, a specified number of times, or until a condition is satisfied, and to provide a means of return to the normal sequence.

Option 1.

```
PERFORM  procedure-name-1 [THRU procedure-name-2]
```

Option 2.

```
PERFORM  procedure-name-1 [THRU procedure-name-2] { data-name-1  } TIMES
                                                  { integer-1    }
```

Option 3.

```
PERFORM  procedure-name-1 [THRU procedure-name-2] UNTIL  condition-1
```

Option 4.

```
PERFORM  procedure-name-1 [THRU procedure-name-2] VARYING data-name-2
         FROM { data-name-3 } BY { data-name-4 } UNTIL  condition-1
              { literal-1   }    { literal-2   }
```

Option 5.

```
PERFORM procedure-name-1 [THRU procedure-name-2] VARYING subscript-name-1
    FROM {data-name-5 | integer-2} BY {data-name-6 | integer-3} UNTIL condition-2
[   AFTER subscript-name-2  FROM {data-name-7 | integer-4} BY {data-name-8 | integer-5}
    UNTIL condition-3 [AFTER subscript-name-3  FROM {data-name-9 | integer-6}
    BY {data-name-10 | integer-7} UNTIL condition-4] ]
```

Notes:

1. PERFORM is the means by which loops are written in COBOL. The loop may be executed once, or a number of times, as determined by a variety of controls.

2. The first statement of procedure-name-1 is the point to which sequence control is sent by PERFORM. The return mechanism is automatically inserted as follows:

   a. If procedure-name-1 is a paragraph-name, and procedure-name-2 is not specified, the return is after the last statement of the procedure-name-1 paragraph.

   b. If procedure-name-1 is a section-name, and procedure-name-2 is not specified, the return is after the last statement of the last paragraph of the procedure-name-1 section.

   c. If procedure-name-2 is specified and is a paragraph-name, the return is after the last statement of the procedure-name-2 paragraph.

   d. If procedure-name-2 is specified and is a section-name, the return is after the last statement of the last paragraph of the procedure-name-2 section.

The last statement performed in all of the above cases must not contain a GO TO verb. The GO TO and PERFORM verbs may occur between procedure-name-1 and the end of procedure-name-2. If there are two or more paths to the end of the loop, procedure-name-2 must be a paragraph consisting of the verb EXIT, to which these paths must lead.

3. In all cases, after the completion of a PERFORM, a bypass is automatically created around the return mechanism which had been inserted after the last statement. Therefore, when no related PERFORM is in progress, sequence control will pass through a last statement to the following statement as if no PERFORM had existed.

4. The PERFORM mechanism for options 1 through 4 operates as follows, with note 3 above applying to all options:

   a. Option 1 is a simple PERFORM. A return to the statement following the PERFORM is inserted after the last statement as defined in note 2, and sequence control is sent to procedure-name-1.

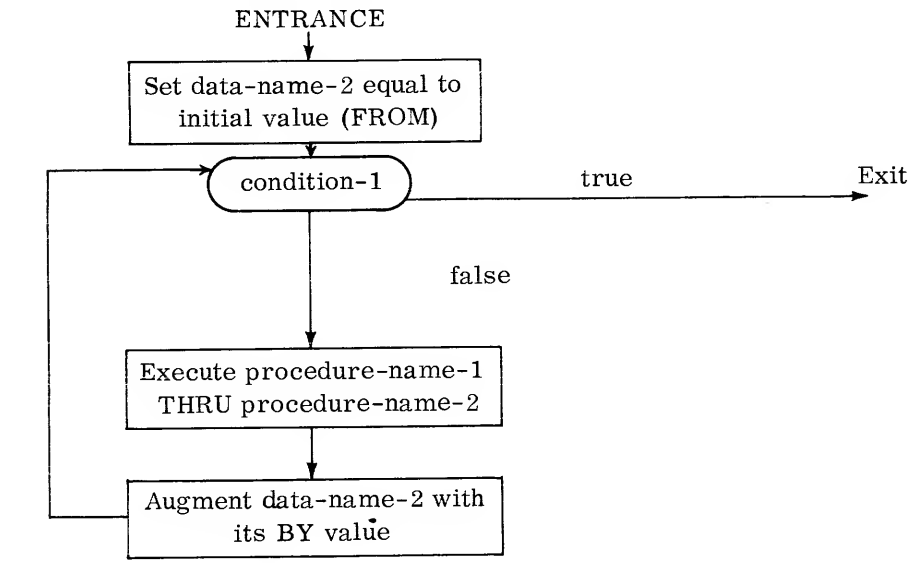
   b. Option 2 is the TIMES option. The specified number of times must be a positive integer, and may be zero. The PERFORM mechanism sets up a counter and tests it against the specified value before each jump to procedure-name-1. The return mechanism after the last statement increases the counter and then sends control to the test. The test gives control to procedure-name-1 the specified number of times, and after the last time sends control to the statement following the PERFORM.

   c. Option 3 is the UNTIL option. This option is the same as the TIMES option, except that an evaluation of a condition takes the place of counting and testing against a specified integer. The condition may be any simple or compound condition, for example, the condition may involve relations and tests. When the condition is satisfied, i.e., when the statement is true, control is transferred to the next statement after the PERFORM statement. If the condition is true when the PERFORM is entered, no jump to procedure-name-1 takes place, and control is transferred to the next statement after the PERFORM statement.

   d. Option 4 is the VARYING data-name option. The VARYING option is assumed to be arithmetic, and all the arithmetic rules apply. This option is used when it is desired to increase or decrease the value of any item while the execution of a procedure or a series of procedures is being accomplished. Only one item can be varied for each PERFORM statement using this option. The PERFORM mechanism sets the value of data-name-2 equal to its starting value (the FROM), then

evaluates the condition (the UNTIL) for truth or falsity. If the condition is true at this point, then no execution of procedure-name-1 through procedure-name-2 takes place. Instead, control is transferred to the next statement after the PERFORM statement. If the condition is false, then procedure-name-1 through procedure-name-2 is executed once. The mechanism then augments the value of data-name-2 by the specified increment or decrement (the BY), and again evaluates the condition (the UNTIL) for truth or falsity. The cycle continues until the condition is determined to be true, at which point control is transferred to the next statement after the PERFORM statement. Literal-1 and literal-2 must be numeric literals, but need not necessarily be integral. A diagram for this mechanism is shown below:

One Subscript



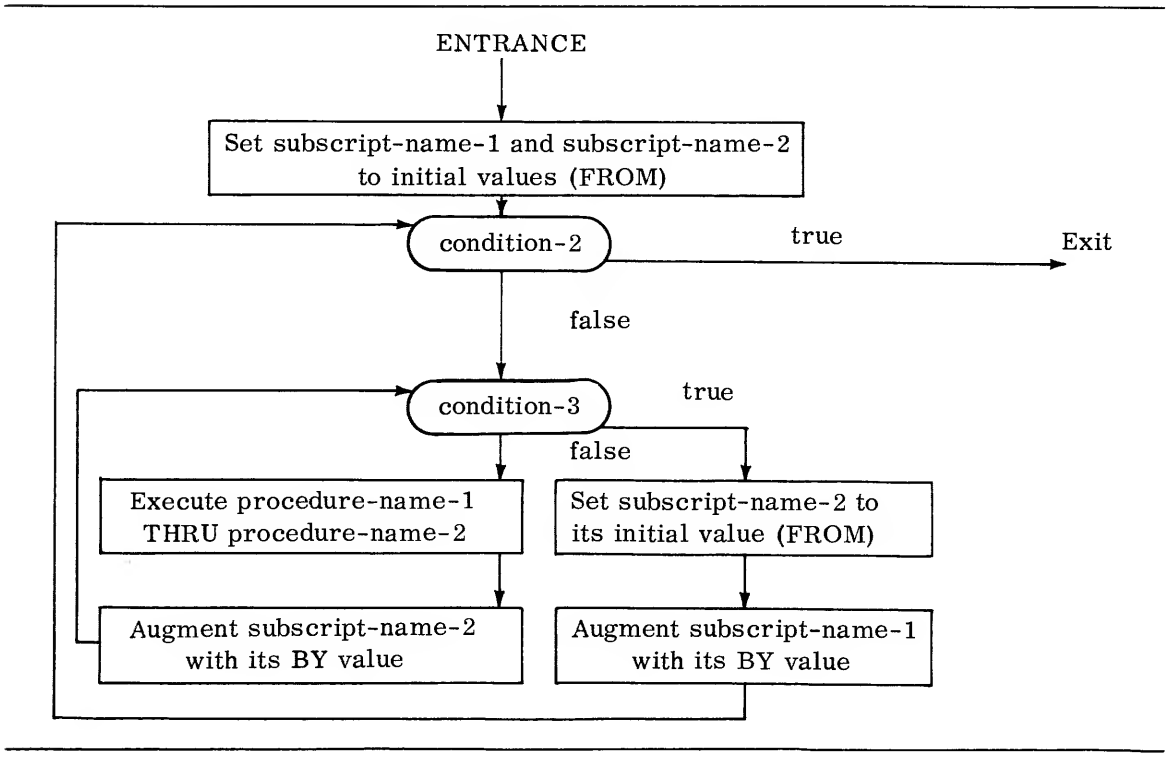
It should be noted that after completion of the PERFORM, data-name-2 will have a value which replaces its previously used value by one increment or decrement, whichever the case may be.

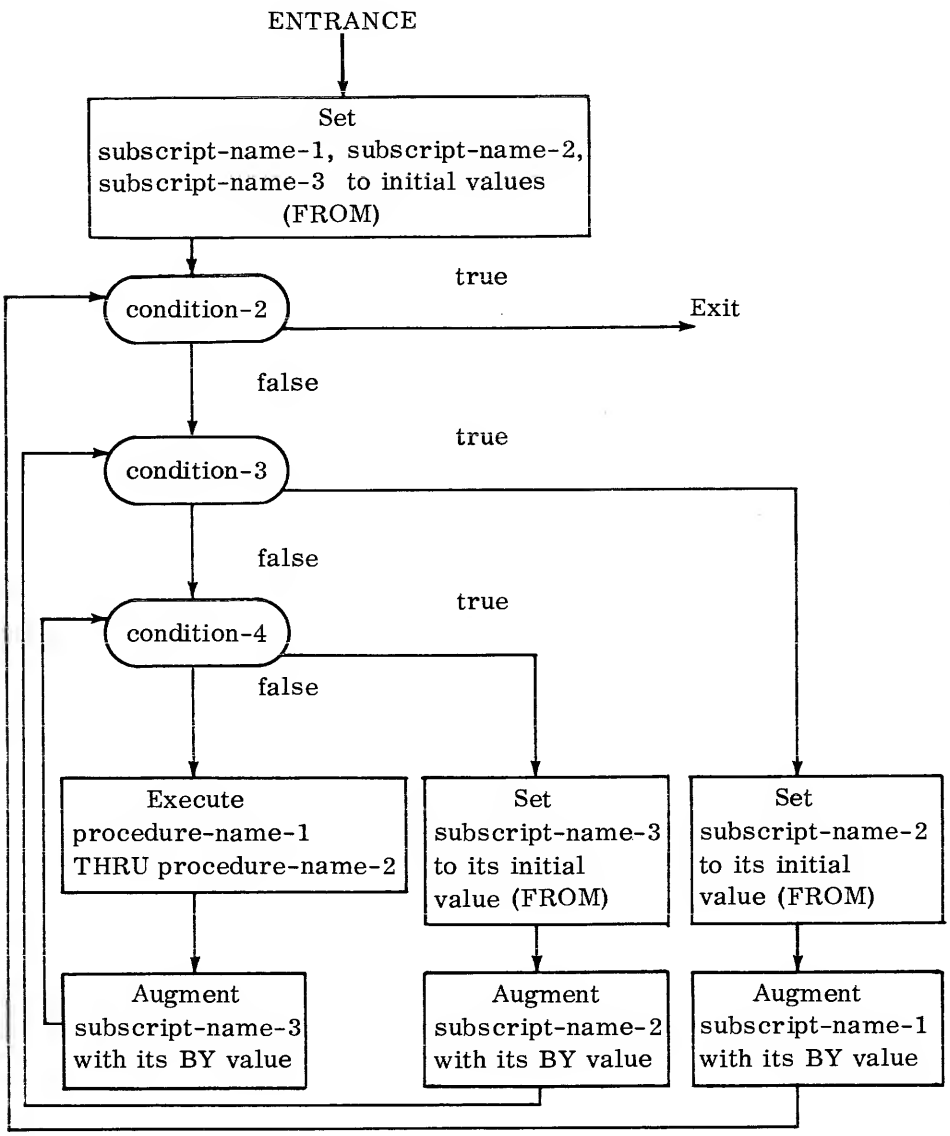
e. Option 5 is the VARYING subscript-name option. This option is used when it is desired to augment the value of one or more subscripts in a nested fashion while the execution of a procedure or a series of procedures is being accomplished. A maximum of three subscripts can be varied per PERFORM statement using this option. When only one subscript is being varied, the mechanism is exactly the same as that of the VARYING data-name-2 option (option 4). When two subscripts are varied, the value of subscript-name-2 goes through a complete cycle (FROM, BY, UNTIL) each time that subscript-name-1 is augmented

with its BY value. The PERFORM is completed as soon as condition-2 is found to be true. When three subscripts are varied the value of subscript-name-3 goes through a complete cycle (FROM, BY, UNTIL) each time that subscript-name-2 is augmented with its BY value. Furthermore, subscript-name-2 goes through a complete cycle (FROM, BY, UNTIL) each time that subscript-name-1 is augmented with its BY value.

The PERFORM is completed as soon as condition-2 is found to be true. Regardless of the number of subscripts being varied, as soon as condition-2 is found to be true, control is transferred to the next statement after the PERFORM statement. The FROM value must be a positive, non-zero integer. The BY value must be a non-zero integer, which may be either positive or negative. Subscript-name-1 subscript-name-2, and subscript-name-3 must never refer to the same item, i.e., they must not be alternate names for the same data item. Diagrams for this mechanism are shown below:

Two Subscripts



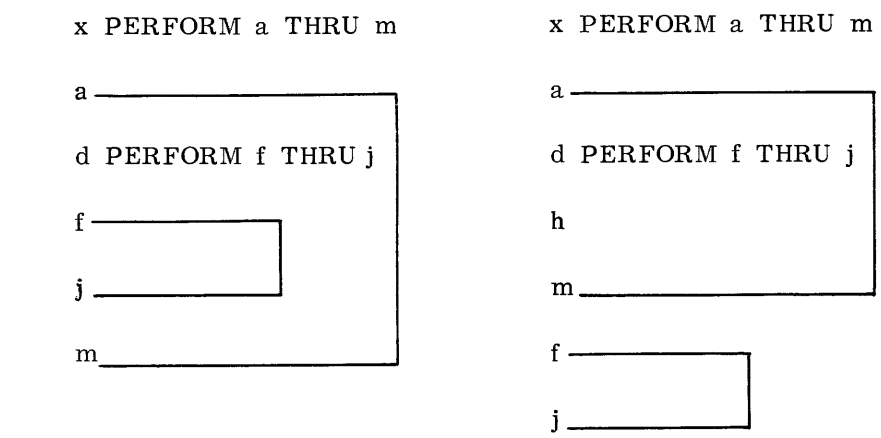


It should be noted that after the completion of the PERFORM, subscript-name-2 and subscript-name-3 will each have a value equal to their respective initial settings (FROM); while subscript-name-1 will have a value which replaces its last used value by one increment or decrement, whichever the case might be.

5. In general, procedure-name-1 should not be the next statement after the PERFORM. If it is, the loop will be executed one more time than was probably intended, because after the PERFORM is satisfied control would go to procedure-name-1 in the normal continuation of the sequence.

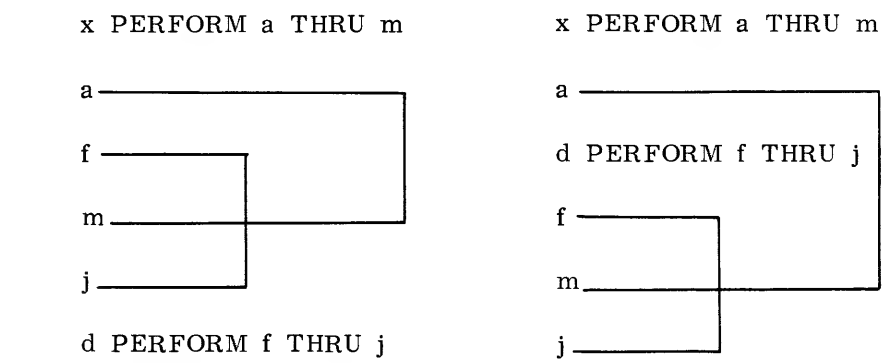
6. If a sequence of statements referred to by a PERFORM includes another PERFORM statement, the sequence associated with the included PERFORM must itself either be totally included in, or totally excluded from the logical sequence referred to by the first PERFORM.

For example, the following illustrations are correct:



The sequence of procedures associated with a PERFORM statement may overlap or intersect the sequence associated with another PERFORM statement, provided that neither sequence includes the PERFORM statement associated with the other sequence.

For example:



READ

Function

To make available the next logical record from an input file and to allow performance of a specified imperative statement when the end of file is detected.

READ file-name RECORD [INTO data-name] , AT END { imperative-statement-1 / NEXT SENTENCE }

[{ ELSE / OTHERWISE } { imperative-statement-2 / NEXT SENTENCE }]

Notes:

1. An OPEN statement for the file must be executed prior to the execution of the first READ for that file.

2. When a file consists of more than one type of logical record, these records automatically share the same storage area. This is equivalent to saying that there exists an implicit redefinition of the area, and only the information which is present in the current record is accessible.

3. No reference can be made by any verb in the Procedure Division to information which is not actually present in the current record. Thus, it is not allowable to refer to the nth occurence of data which appears fewer than n times. If such a reference is made the results in the object program are unpredictable.

4. When the INTO data-name option is used, the data-name must be the name of a Working Storage or output record area. If the format of the data-name differs from that of the input record, moving will be performed according to the rules specified for the MOVE verb without the CORRESPONDING option.

   When the INTO data-name option is used, the file-name RECORD is still available in the input record area.

5. Every READ statement must have an AT END clause.

6. If an OPTIONAL file is not present, the AT END clause will be executed on the first READ. The standard end-of-file procedures will not be performed (see the OPEN verb, and the FILE-CONTROL paragraph in the Environment Division).

7. After execution of the AT END clause an attempt to perform a READ without the execution of a CLOSE and a subsequent OPEN for that file will constitute an error in the object program.

8. After recognition of the end of reel, the READ performs the following operations:

   a. The standard trailer label subroutine.

   b. A tape alternation.

   c. The standard header label subroutine.

   d. Makes the next record available.

STOP

Function

To halt the object program either permanently or temporarily.

```
STOP { literal }
     { RUN   }
```

Notes:

1. If the word RUN is used, then the ending procedure established by the compiler is instituted.

2. If the literal form is used, the literal will be displayed to the operator on the on-line printer.

   Continuation of the object program will begin with the execution of the next statement in sequence.

SUBTRACT

Function

To subtract one, or a sum of two or more, numeric data items from a specified item and set the value of an item equal to the result.

```
SUBTRACT { literal-1 } [ , { literal-2 } ... ]
         { data-name-1 }   { data-name-2 }

FROM { literal-n } [ GIVING data-name-m ] [ ROUNDED ]
     { data-name-n }

[ , ON SIZE ERROR { imperative-statement-1 } ]
                  { NEXT SENTENCE          }

[ { ELSE      } { imperative-statement-2 } ]
  { OTHERWISE } { NEXT SENTENCE          }
```

Notes:

1. All data-names used must refer to the special register, TALLY, or to numeric elementary items whose descriptions appear in the Data Division of the source program.

2. All rules specified under the verb ADD with respect to the ON SIZE ERROR option, the size of operands, the ROUNDED option, the GIVING option, truncation, and the editing of results, apply to the SUBTRACT verb. (See ADD.)

3. If the data description of any literal or data-name used as either the minuend or the subtrahend specifies the presence of an editing symbol, an error will be indicated at compilation time. Operational signs and implied decimal points are not considered editing symbols.

4. When the GIVING option is not used, a literal must not be specified as the minuend.

5. When dealing with multiple subtrahends, the effect of the subtraction will be as if the subtrahends were first added, and the sum was then subtracted from the minuend.

WRITE

Function

To release a logical record for an output file.

WRITE record-name [FROM data-name-1]

Notes:

1. After the WRITE is executed, record-name is no longer available.

2. When the FROM option is used, data-name-1 must be the name of a Working-Storage or an input record area. If the format of data-name-1 differs from that of the record-name, moving will take place according to the rules specified for the MOVE verb (without the CORRESPONDING option). The information in the record-name area is no longer available, but the information in data-name-1 area is available. It is illegal to use the same name for both data-name-1 and record-name.

3. No reference can be made by any verb in the Procedure Division to information which is not actually present in the current record. Thus, referring to the nth occurrence of data which appears fewer than n times is not allowed.

4. An OPEN statement must be executed prior to executing the first WRITE for a file.

5. After recognition of the end of reel, the WRITE performs the following operations:

   a. The standard trailer label subroutine.

   b. A tape alternation.

   c. The standard header label subroutine.

Part VI: REFERENCE FORMAT

This section is included to demonstrate the required layout common to all programs written in COBOL. The rules are few, but binding.

GENERAL DESCRIPTION

The Identification, Environment, Data and Procedure Divisions which constitute a COBOL source program are written on a COBOL Program Sheet in the above order. This reference format, despite its necessary restrictions, is of a relatively free form. The programmer should note, however, that the rules for using it are precise and must be followed exactly, and that these rules take precedence over any other rules with respect to spacing.

PROGRAM IDENTIFICATION CODE (Col. 73-80)

These columns can be used to identify the program using any characters from the COBOL character set, including the blank. The program identification code has no effect on the program.

SEQUENCE NUMBERS (Col. 1 - 6)

The sequence number must consist only of numerals; letters of the alphabet and special characters may not be used. The sequence number has no effect as such on the program and need not be written. However, if the programmer supplies sequence numbers, the compiler will check the source program cards and will indicate any errors in their sequence.

CONTINUATION INDICATOR (Col. 7)

A hyphen in Column 7 indicates that the first character of this line after the indentation is the continuation of the last item written on the preceding line. Regardless of whether the continuation indicator is used or not, continued items must not begin before Margin B (Col. 12) of the subsequent line.

Continuation of Non-Numeric Literals

A non-numeric literal constitutes one COBOL word. Therefore, whenever a non-numeric literal is carried over from one line to another, a hyphen must be placed in Column 7 of the continued line. Remaining spaces on the first line are part of the literal. Leading spaces and a subsequent extra quote mark on the next line are not considered to be part of the literal. The literal resumes immediately after the extra quote mark.

Continuation of Other Words

A hyphen in Column 7 indicates to the compiler that the first character of the continuation is to follow the last character of the preceding line without an intervening space.

If a hyphen is not written in Column 7, it is assumed that the last character of the preceding line is always followed by a blank.

WRITING THE PROGRAM

Columns 8 through 72 are used for the actual data and instructions to be entered into the computer.

Division-Names

The names of divisions must begin at Margin A (Col. 8) followed by a space, the word DIVISION, and then a period. The entry must appear on a line by itself.

Section-Names

A section-name must begin at Margin A followed by a space, the word SECTION, and then a period. The entry must appear on a line by itself.

Other Rules

The rules for writing the division-names and section-names apply to all the divisions. However, a section consists of paragraphs in the Identification, Environment, and Procedure Divisions, whereas it consists of Data Description entries in the Data Division.

Paragraph-Names

A paragraph-name must also begin at Margin A and must be immediately followed by a period and a space. The text may start on the same line. Succeeding lines of the paragraph must begin at Margin B (Col. 12).

Data Description

A File Description entry must begin at Margin A. Succeeding entries may also begin at Margin A, or may be indented to show the data organization. A data-name may not begin before column 12.

ORGANIZATION OF SOURCE PROGRAM

Shown below is a listing of the items which may appear in a source program. Some of the items are absolutely required, while others are optional. This may be checked in the discussion of each individual COBOL word. The order of appearance of the divisions is mandatory. Certain sections within the divisions must also appear as specified, while others have no such rigid rules. Although the sequence given here is the suggested one, unless specifically stated otherwise in the text, this order is not binding.

```
IDENTIFICATION DIVISION.
PROGRAM-ID. program-name.
AUTHOR. author-name.
INSTALLATION. ...
DATE-WRITTEN. ...
DATE-COMPILED. ...
SECURITY. ...
REMARKS. ...


ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. ...
OBJECT-COMPUTER. ...
SPECIAL-NAMES. ...
INPUT-OUTPUT SECTION.
FILE-CONTROL. SELECT ...
I-O-CONTROL. RERUN ...


DATA DIVISION.
FILE SECTION.
FD file-name-1 ...
    01 data-name-1 ...
       02 data-name ...
          03 data-name...
          88 condition-name ...
                :
                :
       02 data-name ...
                :
                :
    01 data-name ...
          :
          :
FD file-name-2 ...
       :
       :
FD file-name-n ...
```

```
WORKING-STORAGE SECTION.
77 data-name ...
   88 condition-name ...
        .
        .
77 data-name ...
     .
     .
01 data-name ...
   02 data-name ...
        .
        .
01 data-name ...
   02 data-name ...
      03 data-name ...
      88 condition-name ...
           .
           .
   02 data-name ...
        .
        .
01 data-name ...
     .
     .

CONSTANT SECTION.
77 data-name ...
     .
     .
77 data-name ...
01 data-name ...
   02 data-name ...
        .
        .
01 data-name ...
   02 data-name ...
      03 data-name ...
           .
           .
   02 data-name ...
        .
        .
01 data-name ...
     .
     .

PROCEDURE DIVISION.
paragraph-name. ...
   .
   .
section-name-1   SECTION.
paragraph-name-1. ...
   .
   .
paragraph-name-2. ...
   .
   .
paragraph-name-n. ...
   .
   .
section-name-2   SECTION.
   .
   .
section-name-n   SECTION.
paragraph-name-1. ...
   .
   .
paragraph-name-2. ...
   .
   .
paragraph-name-n. ...
   .
   .
```

Part VII: COMPLETE LIST OF IBM 7090/7094 COBOL WORDS

The words listed in this section comprise the complete IBM 7090/7094 COBOL vocabulary. Source programmers are warned that these words are pre-empted and may not be used in a COBOL program except as specified in the manual.

Many words are given in both singular and plural form. This is for the convenience of the source programmer, as the compiler will recognize and accept either form.

ABOUT
ACCEPT
ADD
ADDRESS
ADVANCING
AFTER
ALL
ALPHABETIC
ALPHANUMERIC
ALTER
ALTERNATE
AN
AND
APPLY
ARE
AREA
AREAS
AS
ASSIGN
AT
AUTHOR

BCD
BEFORE
BEGINNING
BEGINNING-FILE-LABEL
BEGINNING-TAPE-LABEL
BINARY
BIT
BITS
BLANK
BLANKS
BLOCK
BLOCK-COUNT
BY

CALL
CARD-PUNCH
CARD-READER
CHARACTER
CHARACTERS
CHECK
CHECKPOINT-UNIT
CLASS
CLOCK-UNITS
CLOSE
COBOL
COLLATE-COMMERCIAL
COMPUTATIONAL
COMPUTATIONAL-1
COMPUTATIONAL-2
COMPUTE
CONFIGURATION
CONSTANT
CONTAINS
CONTROL
COPY
CORRESPONDING
CREATION-DATE
CREATION-DAY
CREATION-YEAR

DATA
DATE-COMPILED
DATE-WRITTEN
DECLARATIVES
DEFINE
DENSITY
DEPENDING
DIGIT
DIGITS
DISPLAY
DIVIDE
DIVIDED
DIVISION
DOLLAR

Word List (continued)

ELSE
END
ENDING
ENDING-FILE-LABEL
ENDING-TAPE-LABEL
END-OF-FILE
END-OF-REEL
ENTER
ENVIRONMENT
EQUAL
EQUALS
ERROR
EVERY
EXAMINE
EXCEEDS
EXIT
EXPONENTIATED

FD
FILE
FILE-CONTROL
FILE-IDENTIFICATION
FILE-SERIAL-NUMBER
FILLER
FILLING
FIRST
FLOAT
FOR
FORMAT
FROM

GIVING
GO
GREATER

HASHED
HEADER
HIGH
HIGH-VALUE
HIGH-VALUES

IBM 7090
IBM 7094
ID
IDENTIFICATION
IF
IN
INCLUDE
INPUT
INPUT-OUTPUT
INSTALLATION
INTO
I-O-CONTROL
IS

JUSTIFIED

KEY-S
KEY-1
⋮
KEY-35

LABEL
LABEL-IDENTIFIER
LEADING
LEAVING
LEFT
LESS
LIBRARY
LINES
LINKAGE-MODE
LOCATION
LOCK
LOW
LOW-VALUE
LOW-VALUES
LOWER-BOUND
LOWER-BOUNDS

MEMORY
MEMORY-DUMP
MEMORY-DUMP-KEY
MINUS
MODE
MODULES
MOVE
MULTIPLE
MULTIPLIED
MULTIPLY

NEGATIVE
NEXT
NO
NO-MEMORY-DUMP
NONE
NOT

Word List (continued)

NOTE
NUMERIC

OBJECT-COMPUTER
OBJECT-PROGRAM
OCCURS
OF
OFF
OMITTED
ON
OPEN
OPTIONAL
OPTIONAL-USAGE
OR
OTHERWISE
OUTPUT

PERFORM
PICTURE
PLACE
PLACES
PLUS
POINT
POSITION
POSITIVE
PREPARED
PRINTER
PRIORITY
PROCEDURE
PROCEED
PROGRAM-ID
PROTECT
PROTECTION
PURGE-DATE

QUOTE
QUOTES

RANGE
READ
RECORD
RECORD-COUNT
RECORDING
RECORDS
REDEFINES
REEL
REELS
REEL-NUMBER
REEL-SEQUENCE-NUMBER
REEL-SERIAL-NUMBER
REMARKS
RENAMES
RENAMING
REPLACING
RERUN
RESERVE
RETENTION-PERIOD
RETURNING
REVERSED
REWIND
RIGHT
ROUNDED
RUN

SAME
SECTION
SECURITY
SEGMENT-LIMIT
SELECT
SENTENCE
SENTINEL
SEQUENCED
SIGN
SIGNED
SIZE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STANDARD-LABEL
STATUS
STOP
SUBTRACT
SUPERVISOR
SUPPRESS
SYNCHRONIZED
SYSTEM-INPUT-UNIT
SYSTEM-OUTPUT-UNIT

TALLY
TALLYING
TAPE
TAPE-UNIT

Word List (continued)

TAPE-UNITS
TEST-PATTERN
THAN
THEN
THROUGH } Equivalent
THRU }
TIME
TIMES
TO
TYPE
UNEQUAL
UPPER-BOUND
UPPER-BOUNDS
UNTIL
UPON
USAGE
USE
USING

VALUE
VALUES
VARYING

WHEN
WITH
WORDS
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

Printed in U.S.A. J28-6260-1

IBM®

International Business Machines Corporation
Data Processing Division, 112 East Post Road, White Plains, N.Y.

IBM Technical Newsletter

File Number 7090-24

Re: Form No. J28-6260-1

This Newsletter No. N28-0070

Date December 13, 1963

Previous Newsletter Nos. None

IBM 7090/7094 COBOL

This newsletter contains addenda and errata to the publication, IBM 7090/7094 Programming Systems: COBOL Language, Preliminary Specifications, Form J28-6260-1. It describes features of the 7090/7094 COBOL language which had previously been deferred. Form J28-6260-0 together with Technical Newsletters N28-0040 and N28-0052 are equivalent in information to Form J28-6260-1. This pagination of Form J28-6260-0 differs slightly from that of Form J28-6260-1. This newsletter applies to Form J28-6260-1 and, except for pagination, to Form J28-6260-0.

Page	Amendment
Table of Contents	On the second page of the "Table of Contents" change the heading: EDITING CLAUSES 48 to: BLANK WHEN ZERO 48 On the fourth page of the "Table of Contents": 1. Before the heading "Formulas" add: Declaratives 74 2. Before the heading "EXIT" add: EXAMINE 86 3. Before the heading "WRITE" add: USE 103
6	Replace the first paragraph of note 2 with: 2. Condition-names. A condition-name is the name assigned to a specific value, within the complete set of values that a data-name may assume. A condition-name must contain at least one alphabetic character. The data-name itself is called a conditional variable and the values it may assume are referred to by condition-names. A conditional variable may not be described as a report, scientific decimal, or floating-point item.

International Business Machines Corp., Programming Systems Publications, P. O. Box 390, Poughkeepsie, N. Y.

PRINTED IN U.S.A.

Page	Amendment
9	Change the second sentence under the explanation of "ALL" to read: An alternative form for 'literal' is any figurative constant (not bounded by quotation marks), e.g., ALL SPACES. Add the following to the paragraph after the explanation of "ALL": When non-numeric figurative constants are used in the Procedure Division, they may not be associated with data items whose length is greater than 120 characters. A figurative constant may be specified in a Record Description VALUE clause for an item longer than 120 characters.
12	Change the first sentence to read: When the subscript is represented by data-name, the data-name must be an elementary item described in the Data Division, or may be the name of the special register, TALLY. Replace note 3 at the bottom of the page with: 3. Anywhere within the Data Division itself. For example, ...REDEFINES Y (2) is not permitted.
17	Add the following paragraph after the heading "CONFIGURATION SECTION": For further information on identifying the computer on which a program is to be compiled or run, see the section, "Control Card Discussion" in: IBM 7090/7094 Programming Systems, IBJOB Processor, Form C28-6275-1. Replace the format of the Source-Computer paragraph with: SOURCE COMPUTER. { IBM-7090 / IBM-7094 } .
18	Replace the format of the Object-Computer paragraph with: OBJECT-COMPUTER. { IBM-7090 / IBM-7094 } . Delete notes 2 and 4, changing note 3 to note 2.

Page	Amendment
19	Replace the format of the "SPECIAL-NAMES" paragraph with:

```
SPECIAL-NAMES.  KEY  literal

[IS mnemonic-name-1]

    { , ON STATUS IS condition-name-1  }
    { , OFF STATUS IS condition-name-2 }
    { , ON STATUS IS condition-name-3  }
    { , OFF STATUS IS condition-name-4 }

    [KEY literal ...] .
```

Replace note 1 with the following:

1. This paragraph is not required if the above condition-names are not used in the Procedure Division.

Replace note 2 with the following:

2. This paragraph is used to assign condition-names to the ON and/or OFF status of console switches. The console switches which may be tested are the 36 console entry keys S, 1-35. Literal may be the letter S or one of the numbers 1 through 35, and corresponds to a console entry key. Any of the keys may be designated only once in the Special-Names paragraph. If both ON and OFF are used, the order in which the ON and OFF clauses are specified is not significant. By definition, a switch is ON if it is down and OFF if it is up.

20	Delete the word "OPTIONAL" from the formats for Options 1 and 2.

Place a period at the end of the format for Option 1 of the "FILE-CONTROL" paragraph.

Page | Amendment

Add the following option:

Option 3.

FILE-CONTROL. SELECT file-name-1

[RENAMING file-name-2]

ASSIGN TO HYPERTAPE { hypertape-unit-name-1 FOR MULTIPLE REEL | hypertape-unit-name-2 | hypertape-unit-name-3 FOR MULTIPLE REEL }

[SELECT...] .

---

21 Delete note 2 and renumber the following notes accordingly.

22 Add the following after the third sentence under "Unit Assignment":

A further discussion of unit assignment is contained in: IBM 7090/7094 Programming Systems: IBJOB Processor, Form C28-6275.

In the first chart, add the following, for X, under "Significance of Symbol Shown":

The available units on all channels are tape units only.

Add the following after the chart on "symbolic intersystem units" at the bottom of the page:

In designating an intersystem input unit, model specification should not be made. A R may be added to any symbolic name for an intersystem unit to indicate to the System Monitor that reserve status for the unit is to end after the current job is complete.

23 Add the following before the discussion of "System Units":

c. Symbolic Hypertape Unit Names. Files may be assigned to 7340 Hypertape units with symbolic names of the following form:

XHK/I where:

Page Amendment

Symbolic Name	Interpretation Given to the Name by IBLDR
X	Denotes one of the real channels A, B, ... H.
H	Denotes Hypertape.
K	Denotes one of the unit numbers 1, ..., 9, 0.
I	Denotes interface 0 or 1.

If interface 0 is intended the designation /I may be omitted.

Change "c. System Units." to "d. System Units."

24 Add the following to the chart at the top of the page:

SYSCK1 The system checkpoint unit is assigned to the file.

Add the following under the chart:

7340 Hypertape units may be assigned system names. Specific assignments of this type are controlled by IBSYS control cards. The following restrictions apply:

1. Specification of DENSITY in the RECORDING MODE clause is ignored.

2. If two units have been assigned to a file and the first is a Hypertape unit, the second must also be a Hypertape unit.

Delete the first sentence under note "d. NONE."

Page	Amendment
25	**Replace the page with the following:**

File Assignment Table

Form of ASSIGN TO Clause	Contents of Associated $FILE Card Fields		
	Unit 1 Field	Unit 2 Field	Multireel Field
1 TAPE-UNIT	omitted	omitted	omitted
1 TAPE-UNIT MULTIPLE REEL	omitted	omitted	REELS
2 TAPE-UNITS MULTIPLE REEL	*	omitted	REELS
CARD-READER	CRD	omitted	omitted
CARD-PUNCH	PCH	omitted	omitted
PRINTER	PRT	omitted	omitted
symbolic-tape-unit-name-1	symbolic-tape-unit-name-1	omitted	omitted
Example: A	A	omitted	omitted
symbolic-tape-unit-name-2 MULTIPLE REEL	symbolic-tape-unit-name-2	omitted	REELS
Example: T1V MULTIPLE REEL	T1V	omitted	REELS
symbolic-tape-unit-name-3 symbolic-tape-unit-name-4 FOR MULTIPLE REEL	symbolic-tape-unit-name-3	symbolic-tape-unit-name-4	REELS
Example: C(3), C(4) MULTIPLE REEL	C(3)	C(4)	REELS
symbolic-card-unit-name-1	symbolic-card-unit-name-1	omitted	omitted
Example: RDA	RDA	omitted	omitted
system-unit-name-1	abbreviated system-unit-name-1	omitted or may be filled in automatically	
Example: SYS1N1 #	1N1	IN2	REELS
system-unit-name-2 MULTIPLE REEL	abbreviated system-unit-name-2	omitted or may be filled in automatically	REELS
Example: SYSUT1 MULTIPLE REEL	UT1	omitted	REELS
system-unit-name-3, system-unit-name-4	abbreviated system-unit-name-3	abbreviated system-unit-name-4	REELS
Example: SYSUT2, SYSUT3 FOR MULTIPLE REEL	UT2	UT3	REELS
NONE	NONE	omitted	omitted

Note: In the example, this particular file automatically takes on the MULTIREEL characteristic of the system unit. This is also true for SYSOU1 and SYSPP1 which are system MULTIREEL files.

Page	Amendment

26 Replace the format of the "I-O-CONTROL" paragraph with:

```
[ I-O-CONTROL.                                          ]
[ RERUN  [ON CHECKPOINT-UNIT]  EVERY                    ]
[ BEGINNING OF REEL OF file-name-1                      ]
[ [, file-name-2...]  [RERUN...] .                      ]
```

Add the following note:

4. When the ON CHECKPOINT-UNIT option is not specified, file-name-1, file-name-2, etc., must be the names of output files.

29 In the format add a left bracket before "RECORD CONTAINS."

30 Delete note 1 under "BLOCK SIZE" and renumber the notes which follow accordingly.

31 Replace the third and fourth sentences of the first paragraph of note 8 with the following:

VAR in the table is used to specify a file which contains logical records of different sizes. The logical records may include one or more logical records described by an OCCURS clause with a DEPENDING ON option.

Replace the last paragraph of note 8 with the following:

If a file contains logical records of different sizes, a control word in front of each logical record is assumed when reading or is supplied when writing, unless the WITHOUT COUNT CONTROL option is specified in the RECORDING MODE clause. MAXRECLTH, therefore, equals logical record length plus one. When the CHARACTERS form of the clause is used, the control words must be included in the block size specified.

33 Add the following to the discussion of "BLOCK SIZE":

9. The following chart illustrates the relationship between different types of files, the media on which they are represented, and the presence of control words. Optional signifies that control words may be specified as absent for the file by specifying the WITHOUT COUNT CONTROL option in the RECORDING MODE clause. R stands for the RECORDS form of the clause; C stands for the CHARACTERS form of the clause.

Page Amendment

Type	Use	Blocked	Block Units	Unit Medium	RECORDING MODE Clause	Control Words	File Characteristics
1	Input	No	--	Tape	BCD or BINARY	No	Same Length Records
2	Input	No	--	Card	BCD	No	Same Length Records
3	Input	No	--	System-Unit(3)	BCD or BINARY(1)	No	Same Length Records
4	Output	No	--	Tape	BCD or BINARY	No	Same Length Records
5	Output	No	--	Card	BCD	No	Same Length Records
6	Output	No	--	System-Unit(4)	BCD	No	Same Length Records
7	Input	No	--	Tape	BCD or BINARY	Optional	Differing Record Lengths
8	Input	No	--	Card	BCD	No	Differing Record Lengths
9	Input	No	--	System-Unit(3)	BCD or BINARY(1)	Optional (2)	Differing Record Lengths
10	Output	No	--	Tape	BCD or BINARY	Optional	Differing Record Lengths
11	Output	No	--	Card	BCD	No	Differing Record Lengths
12	Output	No	--	System-Unit(4)	BCD	Optional(2)	Differing Record Lengths
13	Input	Yes	R or C	Tape	BCD or BINARY	No	Same Length Records
14	Output	Yes	R or C	Tape	BCD or BINARY	No	Same Length Records
15	Input	Yes	R or C	Tape	BCD or BINARY	Optional	Variable Length Record(5)
16	Output	Yes	C	Tape	BCD or BINARY	Optional	Variable Length Record(5)
17	Output	Yes	R	Tape	BCD or BINARY	Optional	Variable Length Record(5)
18	Input	Yes	R or C	Tape	BCD or BINARY	Yes	Differing Record Lengths
19	Output	Yes	C	Tape	BCD or BINARY	Yes	Differing Record Lengths
20	Output	Yes	R	Tape	BCD or BINARY	Yes	Differing Record Lengths

(1) Binary recording mode permissible only if unit medium is tape.

(2) Control word permissible only if unit medium is tape.

(3) May only be system unit designated by the name SYSIN1.

(4) May only be system unit designated by the names SYSOU1 or SYSPP1.

(5) One logical record type (only one data-name in the DATA RECORDS clause for the file) containing an OCCURS clause with a DEPENDING ON option in its description.

Page	Amendment
35	Add the following note to the discussion of "LABEL RECORDS":

6. Files that are assigned to SYSIN1, SYSOU1, or SYSPP1 may not be labeled.

36 Replace the format of the "RECORDING MODE" clause with:

$$\left[\text{, RECORDING MODE IS } \left\{\begin{matrix}\underline{\text{BCD}}\\ \underline{\text{BINARY}}\end{matrix}\right\} \left[\left\{\begin{matrix}\underline{\text{LOW}}\\ \underline{\text{HIGH}}\end{matrix}\right\} \text{ DENSITY }\right] \left[\underline{\text{WITHOUT COUNT CONTROL}}\right]\right]$$

---

37 Add the following notes to the discussion of the "RECORDING MODE" clause:

5. When the WITHOUT COUNT CONTROL option is specified, control words preceding logical records on files containing logical records of different sizes will not be assumed when reading and will not be supplied when writing.

   A discussion of buffer size calculations is contained in the discussion of BLOCK SIZE.

Under "Contents of literal" for "FILE-IDENTIFICATION," change the text to read:

Eighteen or less alphanumeric characters which identify the file.

38 Under "Contents of literal" for "FILE-IDENTIFICATION," change the text to read:

Eighteen or less alphanumeric characters which identify the file.

39 Change the last sentence of note 1 to read:

If it is not specified, the first eighteen characters of the file-name are placed in the labels created.

43 Under "COMPLETE ENTRY SKELETON," replace:

[, editing clauses ...]

with:

[, BLANK...]

Page	Amendment

Replace note 2 with the following:

2. Those clauses which begin with SIGNED, SYNCHRONIZED, POINT, PICTURE, and BLANK must not be specified except at an elementary level.

45 In the format for "Report Items," replace [editing clauses (1)] with:

[BLANK WHEN ZERO]

46 Add the following to the format for "Scientific-Decimal Items":

[VALUE IS floating-point-literal]

Delete part d of note 1.

48 Replace the section, "EDITING CLAUSES" with the following:

BLANK WHEN ZERO

Function

To specify that an item will be filled with blanks (spaces) whenever the value of the item is zero.

[, BLANK WHEN ZERO]

Notes:

1. When the value of the item is zero, all editing specifications in a PICTURE clause will be overridden in favor of inserting all spaces.

2. An item is considered a report item when BLANK WHEN ZERO is specified for it. It will be considered CLASS ALPHANUMERIC and USAGE DISPLAY.

3. This clause may only be used at an elementary level.

4. This clause may not be specified when all the numeric character places in the PICTURE clause for an item contain asterisks.

Page	Amendment
50	Replace the format under "OCCURS" with:

```
[, OCCURS integer-1 TIME [S]  [DEPENDING ON data-name]]
```

Replace note 1 with the following:

1. Integer-1 must be a numeric literal with a positive integral value and may not equal zero.

51 Delete notes 5 and 6.

Replace note 7 with the following:

5. The use of the DEPENDING ON option means that the count of the occurrences of the data is equal to the value of the elementary item called data-name. This value must be a positive integer. If the data-name used in the DEPENDING ON option appears within the record in which the current Record Description entry also appears, then data-name must precede the variable portion of the record. In this case, integer-1 is considered as the maximum number of occurrences and is used for storage reservation.

Replace note 8 with the following:

6. Data-name should be qualified when necessary, but subscripting is not permitted. Data-name may be the special register TALLY.

54 Add the following to the paragraph on the "Floating Dollar Sign":

The floating dollar signs must be the leftmost characters in a PICTURE with the exception that commas may be embedded. Suppression of leading commas is also provided.

55 Replace the discussion of scientific decimal items with the following:

c. Scientific Decimal Items.

A scientific decimal item is a special type of report item which specifies editing of a floating-point number. The PICTURE of a scientific decimal item may contain only the following characters:

```
+ - 9 . V E
```

Specifically the PICTURE must conform to the form:

```
{+}   [9(m)]   [{V}]   [9(n)]   E      {+}   99
{-}            [{.}]                   {-}
 Mantissa                              Exponent
```

Page Amendment

where m and n are positive integers and m + n must be greater than 0 and less than 17. The symbol E is used to give visual separation of the exponent from the mantissa.

61 Add the following note to the discussion of the "USAGE" clause:

7. Items specified as USAGE DISPLAY are stored in the 7090/7094 BCD storage code in multiples of six bits. Each six-bit group represents one character in the BCD character mode. COMPUTATIONAL items are stored as binary numbers in the least multiple of six bits capable of holding the item including its sign. The SIZE of a computational item as indicated in the SIZE or PICTURE clause and the number of six-bit groups which it occupies are not, in general, the same. The chart which follows indicates the relationship between the size of a COMPUTATIONAL item as specified in the SIZE or PICTURE clause and the number of six-bit groups the item occupies.

Size of item as indicated in SIZE or PICTURE clause	Number of six-bit groups occupied by item
1	1
2	2
3	2
4	3
5	3
6	4
7	5
8	5
9	6
10	6
11	7
12	7
13	8
14	8
15	9
16	9
17	10
18	11

70 Add the following to the section on "COLLATING SEQUENCE":

The commercial collating sequence is assumed by the Compiler unless the 7090/7094 collating sequence is specified by placing BINSEQ on the $IBCBC card.

Page	Amendment

71 Replace the format for the "Full Relation Test" with the following:

```
                                  ⎧ IS [NOT] GREATER THAN ⎫   ⎧ data-name-2 ⎫
IF ⎧ data-name-1 ⎫                ⎪ IS [NOT] LESS THAN    ⎪   ⎪ literal-2   ⎪
   ⎪ literal-1   ⎪                ⎪ IS [NOT] =            ⎪   ⎩ formula-2   ⎭
   ⎩ formula-1   ⎭                ⎩ IS [NOT] EQUAL TO     ⎭
```

and delete the two sentences that follow.

72 Replace note 3 with the following:

3. Conditional Variable Test. An item whose specific values can be named is called a conditional variable. A name given to a specific value is called a condition-name. Condition-names must be specified with a level number of 88 and must contain a VALUE clause.

   A conditional variable test is one in which a conditional variable is tested to see whether its value is equal to the value specified for a condition-name associated with it.

   The format for the conditional variable test is:

   IF [NOT] condition-name

Replace the format of the "Switch Status Test" with:

IF [NOT] condition-name

74 Add the following before the discussion of "FORMULAS":

DECLARATIVES

Declaratives are procedures that are performed in conjunction with the 7090/7094 Input/Output Control System.

Declaratives, if present, must be grouped together at the beginning of the Procedure Division, and must be preceded by the key word, DECLARATIVES, and followed by the key words END DECLARATIVES.

Each Declarative must constitute a single section headed by a section-name and the key word SECTION. A USE sentence must immediately follow the section header.

Page	Amendment
	The USE sentence identifies the conditions calling for the execution of the procedures specified in the paragraphs which follow the USE sentence.
75	Change the last sentence to read:
	Exponentiation of a negative variable or literal is allowed only if the exponent is a literal or data-name having an integral value. If the exponent is other than a non-negative integer, the result of the exponentiation will be zero.
76	Add "USE" to the list of "Compiler Directing Verbs."
80	Change the sentence under 2a, "Input Files," which begins "Note that any label....", to read:
	Any label processing and procedures specified in a USE Declarative are performed only when the physical end of file is encountered on the tape.
	Change the first sentence under 2b, "Output Files," to read:
	The final closing conventions and procedures specified in a USE Declarative are performed and the data area is released.
	Change the first sentence under 3a, "Input Files," to read:
	Processing of the end label will be bypassed, but procedures for checking the label on the next reel, including procedures specified in a USE Declarative, will be executed.
81	Change the first sentence under 3b, "Output Files," to read:
	The standard end-of-reel processing, including execution of procedures specified in a USE Declarative, takes place immediately.
	Delete note 5 and renumber note 6 accordingly.
82	Replace the first line of the format with the following:

$$\underline{\text{COMPUTE}} \quad \text{data-name-1} \quad \left[\underline{\text{ROUNDED}}\right] \quad = \quad \left\{\begin{matrix}\text{data-name-2} \\ \text{formula}\end{matrix}\right\}$$

Delete note 6 and renumber the notes which follow accordingly.

Page	Amendment
83	Replace the format and note 1 of the DISPLAY statement with the following:

```
DISPLAY  { literal-1   }  [ , { literal-2   } ... ]
         { data-name-1 }     { data-name-2 }

         [UPON SYSOU1]
```

Notes:

1. The standard display device is the on-line printer. If the UPON option is specified, the display device will be the system output unit. Automatic carriage control for off-line printing of output on the system output unit is provided. For further information concerning SYSOU1, see "FILE-CONTROL" in the Environment Division.

Delete the last sentence in note 2.

Add the following note to the discussion of "DISPLAY":

6. The special register TALLY may be used as an operand in a DISPLAY statement.

85 Replace the discussion of "ENTER" with:

ENTER

Function

To permit communication between a COBOL object program and one or more subroutines assembled by IBMAP.

Option 1.

```
ENTER LINKAGE-MODE.

CALL 'entry-name'

[ , USING  { data-name-1... }  [ { data-name-m } ]
           { file-name-1...  }  [ { file-name-n } ]

     [, data-name-(m+1)...]  [, data-name-K] ]

[ , RETURNING procedure-name-1 [procedure-name-2...] ].

ENTER COBOL.
```

Option 2.

ENTER LINKAGE-MODE.

ENTRY POINT IS 'entry-name'

[RECEIVE data-name-1... [,data-name-m]]

[PROVIDE data-name-(m+1)... [,data-name-K]].

ENTER COBOL.

Option 3.

ENTER LINKAGE-MODE.

RETURN VIA 'entry-name'

[, DEPENDING ON data-name] .

ENTER COBOL.

Notes:

1. The ENTER LINKAGE-MODE statement is used to provide linkage between programs. After ENTER LINKAGE-MODE is specified only the statements illustrated in the format may be used until ENTER COBOL is specified.

2. ENTER LINKAGE-MODE and ENTER COBOL must have a paragraph-name in the A-margin.

3. Reference format rules for writing statements following ENTER LINKAGE-MODE are the same as those for ordinary COBOL statements.

4. If an ENTER option is repeated or if several options are used consecutively, ENTER LINKAGE-MODE need not be repeated.

5. Option 1 is used in the main program to link to a subordinate program. Options 2 and 3 are used in a subordinate program to link to the main program. When option 2 is used 'entry-name' must be identical to an 'entry-name' in a CALL statement in the main program. When option 2 and option 3 are used in the same subordinate program in conjunction with the source CALL statement, the 'entry-name' in options 2 and 3 and in the associated CALL statement must be identical.

Page Amendment

6. 'Entry-name' defines the entry point of the subordinate program. 'Entry-name' must consist of one to six characters at least one of which must be alphabetic and none of which may be blank. It must be surrounded by quotation marks so that the Compiler will treat it as a non-numeric literal. This allows for the use of names permissible in MAP but not normally permissible in COBOL.

7. If the subordinate program was separately assembled under IBMAP for inclusion with a COBOL program at load time, 'entry-name' in the main COBOL program must correspond to a name specified by a MAP language ENTRY pseudo-operation.

8. The USING option provides a means of informing a subordinate program of the object-time locations of data items or of File Control Blocks.

9. Normal return from the subordinate program is to the next sentence in the main COBOL program or to the next ENTER option, if specified. The RETURNING option is used together with option 3 to provide alternate returns.

10. A CALL statement of the following form:

    CALL 'name' USING X1, X2,...,XA

    will produce a MAP language CALL operation of the following form:

    CALL name (Y1, Y2,...,YA).

    If X_i in the CALL statement is a data-name, the corresponding Y_i will be a computer word which object-time initialization will convert to the form:

    PZE A_i,,B_i

    where A_i is the address of the first word containing X_i and B_i is the number (0 through 5) of the first six-bit group (byte) in the first word containing X_i.

    If X_i is a file-name, the corresponding Y_i will be the address of the first word of the File Control Block for the file. A further discussion of File Control Blocks is contained in the publication, IBM 709/7090 Input/Output Control System, Form C28-6100-2.

11. The RECEIVE option is used in the subordinate program to move information from the main program to the subordinate program. Data-name-1 through data-name-m following RECEIVE are the names of items into which information is stored. Storing takes place when control is transferred to the subordinate program by means of a CALL statement in the main program.

Page Amendment

12. The PROVIDE option is used in the subordinate program to return information to the main program. Data-name-(m+1) through data-name-K following PROVIDE are the names of items whose information is stored into the main program. Storing takes place when control is transferred to the main program by means of option 3.

13. The data-names used with options 1 and 2 must conform to the following rules:

    a. They may not specify items which vary in length at object time.

    b. They may not be report or scientific decimal items.

    c. The total number of data-names specified in the RECEIVE option plus the number of data-names in the PROVIDE option must equal the total number of data-names specified in an associated USING option (that is, where options 1 and 2 have the same 'entry-name'.) Any repetitions of data-names must be included in the total number.

    d. The names of the data items in the associated USING option do not have to be identical to the names of the data items in the RECEIVE and PROVIDE options. However, a data item in the USING clause must be of the same length and description as its counterpart in the RECEIVE or PROVIDE option.

    e. The order of the data items in the associated USING option must correspond to the order of its counterparts in the RECEIVE and PROVIDE options.

14. Option 3 is used in the subordinate program to return to the main program. 'Entry-name' in the RETURN option must be the same as 'entry-name' in the CALL statement in the main program.

15. When the DEPENDING ON option is specified, the value of data-name is equivalent to the number of an alternate return specified in the RETURNING option of an associated CALL statement in the main program. Thus, if the value of data-name is zero, the normal return will be taken; if the value of data-name is 1, return will be to procedure-name-1 specified in the RETURNING option.

16. Following is a coding example illustrating the use of option 1 in a main COBOL program and option 2 and option 3 in a subordinate COBOL program.

Page   Amendment

Main Program

```
PARAGRAPH-NAME-1.  ENTER LINKAGE-MODE.
     CALL 'ENTPNT' USING WORK, WORK, AMOUNT.
PARAGRAPH-NAME-2.  ENTER COBOL.
```

Subordinate Program

```
PARAGRAPH-NAME-3.  ENTER LINKAGE-MODE.
     ENTRY POINT IS 'ENTPNT'
     RECEIVE WORK-A
     PROVIDE WORK-A, AMOUNT-A.
PARAGRAPH-NAME-4.  ENTER COBOL.
          .
          .
          .
PARAGRAPH-NAME-5.  ENTER LINKAGE-MODE.
     RETURN VIA 'ENTPNT'.
PARAGRAPH-NAME-6.  ENTER COBOL.
```

86   Add the following after the discussion of "ENTER":

EXAMINE

Function

To replace certain occurrences of a specified character and/or to count the number of such occurrences in a data item.

```
                        { TALLYING   { ALL         }                        }
                        {            { LEADING     }                        }
                        {            { UNTIL FIRST }                        }
                        {                                                   }
                        {      literal-1 [REPLACING BY literal-2]           }
EXAMINE data-name       {                                                   }
                        { REPLACING  { ALL           }                      }
                        {            { LEADING       }                      }
                        {            { [UNTIL] FIRST }                      }
                        {                                                   }
                        {                 literal-3 BY literal-4            }
```

Notes:

1. Any literal used in an EXAMINE statement must be a member of the character set associated with the CLASS specified for data-name. A figurative constant may be specified and will be considered as a single-character literal. The EXAMINE verb can be applied only to a data item whose USAGE is DISPLAY.

Page Amendment

2. When an EXAMINE statement is executed, the examination begins with the leftmost character of the data item and proceeds to the right. If the CLASS of the item being examined is NUMERIC, any operational sign associated with the item will not be considered when the item is examined.

3. When the TALLYING option is used, a count at object time of the number of certain characters in data-name is made, and this count replaces the value of the special register TALLY. TALLY is the name of a special register (USAGE COMPUTATIONAL SYNCHRONIZED RIGHT) whose length is equivalent to an integer of five decimal digits. Its primary use is to hold information produced by the EXAMINE verb. It may also be used as a data-name in other procedural statements. The count at object time depends on which of the three options of TALLYING is employed:

   a. If ALL is specified, all occurrences of literal-1 in the data item are counted.

   b. If LEADING is specified, the count represents the number of occurrences of literal-1 prior to encountering a character other than literal-1.

   c. If UNTIL FIRST is specified, the count represents the number of characters other than literal-1 encountered prior to the first occurrence of literal-1.

4. When the REPLACING option is used (either with or without the TALLYING option), the replacement of characters depends on which of the four options of REPLACING is employed:

   a. If ALL is specified, literal-2 (or literal-4) is substituted for each occurrence of literal-1 (or literal-3).

   b. If LEADING is specified, the substitution of literal-2 for literal-1 (or literal-4 for literal-3) terminates when a character other than literal-1 (or literal-3) is encountered.

   c. If UNTIL FIRST is specified, the substitution of literal-2 (or literal-4) terminates as soon as the first literal-1 (or literal-3) is encountered.

   d. If FIRST is specified, only the first occurrence of literal-3 is replaced by literal-4.

89 In the chart, for the case where Source Field Type is ALL and Receiving Field Type is External Decimal, replace "yes" with:

yes(6)

Page	Amendment

Add the following note below the chart:

(6) The literal following ALL must be numeric.

93 Change note 5 to read:

5. OPEN initiates the standard IOCS label checking for input and output files and label writing for output files. It is at this time that label-handling procedures specified by a USE Declarative are executed.

94 Delete note 6 under "OPEN."

Replace the format of option 2 under "PERFORM" with:

Option 2.

PERFORM procedure-name-1 [THRU procedure-name-2]

{data-name-1 / integer-1} TIME [S]

101 Delete note 6 and renumber the notes which follow accordingly:

Change note 8 to read:

7. After recognition of the end of reel, the READ performs the following operations:

   a. The standard trailer label subroutine and procedures specified by a USE Declarative.

   b. A tape alternation.

   c. The standard header label subroutine and procedures specified by a USE Declarative.

   d. Makes the next record available.

103 Add the following before the section on "WRITE":

USE

Function

To specify that procedures for input/output error and/or label handling are to be used in addition to procedures supplied by the 7090/7094 Input/Output Control System.

Page Amendment

Option 1.

USE AFTER STANDARD ERROR PROCEDURE

$$\text{ON} \left\{ \begin{array}{l} \text{INPUT} \\ \text{file-name-1} \quad \left[, \text{file-name-2} \ldots \right] \end{array} \right\} .$$

Option 2.

$$\text{USE} \left\{ \begin{array}{l} \text{BEFORE} \\ \text{AFTER} \end{array} \right\} \text{STANDARD} \left[\left\{ \begin{array}{l} \text{BEGINNING} \\ \text{ENDING} \end{array} \right\} \right]$$

$$\left[\left\{ \begin{array}{l} \text{REEL} \\ \text{FILE} \end{array} \right\} \right] \text{LABEL PROCEDURE}$$

$$\text{ON} \left\{ \begin{array}{l} \text{INPUT} \\ \text{OUTPUT} \\ \text{file-name-1} \quad \left[, \text{file-name-2} \ldots \right] \end{array} \right\} .$$

Notes:

1. A USE sentence, when present, must immediately follow a section header within the Declaratives portion of the Procedure Division. Each USE sentence must be followed by one or more procedure paragraphs. The presence of another section or the statement END DECLARATIVES terminates a USE section.

2. The procedure paragraphs that follow a USE sentence may not include statements containing input/output verbs. They may not include statements which transfer control to procedures containing input/output verbs.

3. When option 1 is specified, the procedure paragraphs which follow the USE sentence will be executed after the standard IOCS error routine is completed. Option 1 may only be used in connection with the input files.

4. When option 2 is specified, the procedure paragraphs which follow the USE sentence will be executed before or after the IOCS header or trailer label checking procedure is completed, depending on whether the BEFORE or AFTER option is specified. Option 2 may only be specified for labeled files.

Page Amendment

5. When the BEGINNING option is specified, the procedure paragraphs which follow the USE sentence will be executed for header labels; when the ENDING option is specified, the procedure paragraphs will be executed for trailer labels. If neither BEGINNING nor ENDING is specified, the procedure paragraphs will be executed for both header and trailer labels.

6. When the REEL option is specified, the procedure paragraphs are executed for labels between the first header and the last trailer label of the file. When the FILE option is specified, the procedure paragraphs are executed for the first header and/or the last trailer label of the file. Specifying neither REEL nor FILE is equivalent to specifying both REEL and FILE.

7. When INPUT is specified, the procedure paragraphs are executed for all labeled input files; when OUTPUT is specified, the procedure paragraphs are executed for all labeled output files.

8. When option 2 is specified, external names are generated by the Compiler and placed in the object program Control Dictionary. In jobs consisting of more than one source program, it may, therefore, be necessary to rename non-unique external names by means of $NAME cards. A further discussion of $NAME cards is contained in the publication, IBM 7090/7094 Programming Systems: IBJOB Processor, Form C28-6275-0.

104 Change note 5 to read:

5. After recognition of the end of reel, the WRITE performs the following operations:

   a. The standard trailer subroutine and procedures specified by a USE Declarative.

   b. A tape alternation.

   c. The standard header label subroutine and procedures specified by a USE Declarative.

108 Replace the line "PROCEDURE DIVISION" and what follows with:

```
PROCEDURE DIVISION.
DECLARATIVES.
section-name-1  SECTION.  USE... .
paragraph-name-1. ...
    :
    :
paragraph-name-n. ...
    :
    :
```

Page Amendment

```
section-name-n  SECTION.  USE... .
paragraph-name-1. ...
     :
     :
paragraph-name-n. ...
     :
     :
END DECLARATIVES.
paragraph-name. ...
     :
     :
section-name-1  SECTION.
paragraph-name-1. ...
     :
     :
paragraph-name-2. ...
     :
     :
paragraph-name-n. ...
     :
     :
section-name-2  SECTION.
     :
     :
section-name-n  SECTION.
paragraph-name-1. ...
     :
     :
paragraph-name-2. ...
     :
     :
paragraph-name-n. ...
     :
     :
```

109-112 Replace the word list with the following:

ADD	BLANK
AFTER	BLOCK
ALL	BY
ALPHABETIC	
ALPHANUMERIC	CALL
ALTER	CARD-PUNCH
AN	CARD-READER
AND	CHARACTER
ARE	CHARACTERS
ASSIGN	CHECKPOINT-UNIT
AT	CLASS
	CLOSE
BCD	COBOL
BEFORE	*COLLATE-COMMERCIAL
BEGINNING	COMPUTATIONAL
BINARY	COMPUTATIONAL-1

* This word is not described in this publication but, nevertheless, is pre-empted and should not be used in a 7090/7094 COBOL program.

Page Amendment

COMPUTATIONAL-2
COMPUTE
CONFIGURATION
CONSTANT
CONTAINS
COUNT
CONTROL
CORRESPONDING

DATA
DECLARATIVES
DENSITY
DEPENDING
DIGIT
DIGITS
DISPLAY
DIVIDE
DIVISION

ELSE
END
*ENDING
ENTER
ENTRY
ENVIRONMENT
EQUAL
EQUALS
ERROR
EVERY
EXAMINE
EXCEEDS
EXIT

FD
FILE
FILE-CONTROL
FILE-IDENTIFICATION
FILE-SERIAL-NUMBER
FILLER
FIRST
FOR
FROM

GIVING
GO
GREATER

HIGH
HIGH-VALUE
HIGH-VALUES
HYPERTAPE

IBM-7090
IBM-7094
*IBM7090
*IBM7094
ID
IDENTIFICATION
IF
IN
INPUT
INPUT-OUTPUT
INTO
I-O-CONTROL
IS

KEY

LABEL
LEADING
LEAVING
LEFT
LESS
LINKAGE-MODE
LOCATION
LOCK
LOW
LOW-VALUE
LOW-VALUES

*MINUS
MODE
MOVE
MULTIPLE
MULTIPLY

NEGATIVE
NEXT
NO
NONE
NOT
NOTE
NUMERIC

---

* This word is not described in this publication but, nevertheless, is pre-empted and should not be used in a 7090/7094 COBOL program.

Page Amendment

OBJECT-COMPUTER
OCCURS
OF
OFF
OMITTED
ON
OPEN
*OPTIONAL
OR
OTHERWISE
OUTPUT

PERFORM
PICTURE
PLACE
PLACES
*PLUS
POINT
POSITIVE
PRINTER
PROCEDURE
PROCEED
PROVIDE

QUOTE
QUOTES

READ
RECEIVE
RECORD
RECORDING
RECORDS
REDEFINES
REEL
REEL-SEQUENCE-NUMBER
RENAMING
REPLACING
RERUN
RETENTION-PERIOD
RETURN
RETURNING
REWIND
RIGHT
ROUNDED
RUN

SECTION
SELECT
SENTENCE
SIGNED
SIZE
SOURCE-COMPUTER
SPACE
SPACES
SPECIAL-NAMES
STANDARD
STATUS
STOP
SUBTRACT
SYNCHRONIZED
SYSOU1

TALLY
TALLYING
TAPE-UNIT
TAPE-UNITS
THAN
THEN
THROUGH } Equivalent
THRU }
TIME
TIMES
TO

UNEQUAL
UNTIL
UPON
USAGE
USE
USING

VALUE
VARYING
VIA

WITHOUT
WORKING-STORAGE
WRITE

ZERO
ZEROES
ZEROS

* This word is not described in this publication but, nevertheless, is pre-empted and should not be used in a 7090/7094 COBOL program.

IBM Technical Newsletter

System	7090-24
Re: Form No.	J28-6260-1
This Newsletter No.	N28-0092-1
Date	April 8, 1964
Previous Newsletter Nos.	N28-0070
	N28-0092

IBM 7090/7094 COBOL

This newsletter contains addenda and errata to the publication IBM 7090/7094 Programming Systems: COBOL Language, Preliminary Specifications, Form J28-6260-1. It makes Technical Newsletter N28-0092 obsolete. Form J28-6260-0 together with Technical Newsletters N28-0040 and N28-0052 are equivalent in information to Form J28-6260-1. The pagination of Form J28-6260-0 differs slightly from that of Form J28-6260-1. The pagination in this newsletter and in Technical Newsletter N28-0070 refers to Form J28-6260-1.

Page	Amendment
Table of Contents	On the fourth page of the Table of Contents, before the heading "ADD" add:

ACCEPT 77

26 Replace the format of the "I-O-CONTROL" paragraph with:

I-O-CONTROL.

RERUN [ON CHECKPOINT-UNIT] EVERY
BEGINNING OF REEL OF file-name-1
[, file-name-2...] [RERUN...] .

Replace note 4 with the following:

4. When the ON CHECKPOINT-UNIT option is not specified, file-name-1, file-name-2, etc., must be the names of labeled output files.

50 Add the following sentence to note 3 under LEVEL-NUMBER:

The 01 level number must be used for the superior levels of data organization in the Record Description entries. The first entry must have a level number of 01.

International Business Machines Corp., Applied Programming Publications, 1271 Avenue of the Americas, N. Y. 20, N. Y.

PRINTED IN U.S.A.

Page	Amendment
	Replace note 2 under OCCURS with the following: 2. The OCCURS clause cannot be specified for an entry which has a level number of 01 or 77.
54	Replace the discussions under "The Floating Dollar Sign" and "The Floating Minus Sign" with the following: The Floating Dollar Sign. Zero suppression with a floating dollar sign is specified by placing a dollar sign in each numeric character position to be suppressed. A dollar sign will be placed in the rightmost position in which suppression is to occur. All floating dollar signs must be the leftmost characters in a PICTURE with the exception that single commas, B's, or 0's may be embedded. Suppression of leading commas, B's, and 0's is also provided. The Floating Minus Sign. Zero suppression with a floating minus sign is specified by placing a minus sign in each numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur. If the value is positive or zero, a space will be inserted instead of a minus sign. All floating minus signs must be the leftmost characters in a PICTURE with the exception that single commas, B's, or 0's may be embedded. Suppresion of leading commas, B's, and 0's is also provided.
55	Replace the discussion under "The Floating Plus Sign" with the following: The Floating Plus Sign. Zero suppression by means of a floating plus sign is specified by placing a plus sign in each leading numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur. If the value of the item is not negative, a plus sign will be inserted instead. All floating plus signs must be the leftmost characters in a PICTURE except that single commas, B's, or 0's may be embedded. Suppression of leading commas, B's, and 0's is also provided.
58	Delete note 3 under "SIGNED" and renumber the notes which follow accordingly.
60	Insert the following sentences after the second sentence under "Working-Storage Records." The OCCURS clause may not be used at the 01 level in either Record Descriptions or Working-Storage Record Descriptions. The 01 level number must be used for the superior levels of data organization.

Page	Amendment
69	Change the first sentence under "Simple Conditions" to read:

Simple conditions test for one of five types of conditions.

72 Add as note 5 the following:

5. Class Test

The format for the class test is:

IF data-name IS [NOT] { NUMERIC / ALPHABETIC }

Data-name must be either a group item or an alphanumeric non-report item. When a class test is performed, determination is made as to whether or not:

1. An item consists of the characters 0-9 (NUMERIC).
2. An item consists of the characters A-Z or blank (ALPHABETIC).

When a single-character item contains an operational sign, it will be considered NUMERIC if the test is for a NUMERIC item and ALPHABETIC if the test is for an ALPHABETIC item. Items consisting of more than one character and containing an operational sign on the low-order character will be considered NUMERIC only if all the other characters in the item are numbers.

76 Add ACCEPT to the list of input/output verbs.

77 Add the following before the discussion of ADD:

ACCEPT

Function

To obtain low-volume data from an input device.

ACCEPT data-name

Notes:

1. The standard device from which low-volume data is obtained is the on-line card reader.

Page	Amendment

2. Data-name must be the name of an item defined in the Working-Storage Section. Data-name may not be described with an OCCURS DEPENDING ON clause and may not be followed by any entry within the logical record containing data-name which is described by an OCCURS DEPENDING ON clause. No assumption should be made about the contents of data-name beyond the first 72 characters.

3. Data-name must be USAGE DISPLAY and may only be an alphanumeric or alphabetic non-report item, an external decimal item, or a group item.

4. No editing or error-checking of the value of data-name is performed.

109-112 Add the following words to the COBOL word list as amended by Technical Newsletter N28-0070:

AUTHOR
DATE-COMPILED
DATE-WRITTEN
INSTALLATION
PROGRAM-ID
REMARKS
SECURITY

IBM **Technical Newsletter**

File Number	7090-24
Re: Form No.	J28-6260-2
This Newsletter No.	N28-0103
Date	June 15, 1964
Previous Newsletter Nos.	None

IBM 7090/7094 COBOL

This newsletter supplements the publication IBM 7090/7094 Programming Systems: COBOL Language: Preliminary Specifications, Form J28-6260-2.

Form J28-6260-1 with Technical Newsletters N28-0070 and N28-0092 is equivalent in information to Form J28-6260-2. The pagination in this newsletter refers to Form J28-6260-2.

In the subject publication, replace the pages listed below with the pages that are attached to this newsletter.

1. Pages 7 and 8
2. Pages 9 and 10
3. Pages 25 and 26
4. Pages 39 and 40
5. Pages 41 and 42
6. Pages 53 and 54
7. Pages 55 and 56
8. Pages 71 and 72
9. Pages 77 and 78
10. Pages 79 and 80
11. Pages 81 and 82
12. Pages 93 and 94
13. Pages 95 and 96
14. Pages 107 and 108
15. Pages 109 and 110

A vertical line immediately to the left of the column shows where the text was changed.

File this cover page at the back of the bulletin. It will provide a reference to changes, a method of determining that all amendments have been received, and a check for determining if the bulletin contains the proper pages.

International Business Machines Corp., Programming Systems Publications, P. O. Box 390, Poughkeepsie, N. Y. 12602

PRINTED IN U.S.A.

Each condition-name must be unique, or be made unique through qualification. A conditional variable may be used as a qualifier for any of its condition-names.

If a condition-name is to be equated to the status of a hardware device, it is defined in the SPECIAL-NAMES paragraph of the Environment Division.

The difference between this kind of condition-name and a Data Division condition-name is automatically handled by the compiler.

3. Procedure-names. A procedure-name is either a paragraph-name or a section-name. Procedure-names permit one procedure to refer to others. A procedure-name may be composed solely of numeric characters. However, two numeric procedure-names are equivalent only if they are composed of the same number of numeric digits and have the same numeric value. Thus, 0023 is not equivalent to 23.

4. Literals. A literal is an item of data, integral to the text, which is completely defined by its own identity rather than by Data Division clauses. A literal may belong to one of two classes: non-numeric (alphabetic or alphanumeric) or numeric. Non-numeric literals must be bounded by quotation marks; numeric literals must not.

   A non-numeric literal is defined as a literal which is composed of up to 120 of any allowable characters except the quotation mark. All spaces which are enclosed in the quotation marks are included as spaces in the literal.

   A numeric literal is defined as one which is composed only of characters chosen from the numerals 0 through 9, the plus or minus sign, and the decimal point. The rules for formation of numeric literals are:

   a. A numeric literal may contain only one sign character and/or one decimal point.

   b. The literal must contain at least one and not more than 18 digits.

   c. The sign in the literal must appear as the left-most character. If the literal is unsigned, it is considered to be positive.

   d. The decimal point may appear anywhere within the literal except as the right-most character, and is treated as an implied decimal point. If the literal contains no decimal point, it is considered to be an integer.

(See Record Description Entry VALUE clause for a discussion of floating point literals.)

If a literal conforms to the rules for formation of numeric literals, but it is enclosed in quotation marks, it is considered as a non-numeric literal and will be treated as such by the compiler. For example, -125.65 is not the same as '-125.65'.

Examples of non-numeric literals are:

'EXAMINE CLOCK NUMBER'
'PAGE 144 MISSING'
'-125.65'

Examples of numeric literals are:

1506798
-12572.6
+256.75
1435.89

5. Figurative Constants. Certain values have been assigned fixed data-names. These items with the fixed data-names are called "figurative constants." These names, when used as figurative constants, must not be bounded by quotation marks. The singular and plural forms of figurative constants are equivalent, and may be used interchangeably. If the names are bounded by quotation marks they will be considered as non-numeric literals. The fixed data-names and their meanings are as follows:

ZERO ZEROS ZEROES	Represents the value 0.
SPACE SPACES	Represents one or more blanks or spaces.
HIGH-VALUE HIGH-VALUES	Usually represent one or more 9's, the highest value in the commercial collating sequence; but represent one or more left parentheses if the scientific (binary) collating sequence has been specified by the option BINSEQ on the $IBCBC control card.
LOW-VALUE LOW-VALUES	Usually represent one or more blanks or spaces, the lowest value in the commercial collating sequence; but represent one or more zeros if the scientific (binary) collating sequence has been specified by the option BINSEQ on the $IBCBC control card.
QUOTE QUOTES	Represents the character '. Note that the use of the word QUOTE to represent the character ' at object time is not equivalent to the use of the symbol ' to bound a literal.

ALL 'literal' — Represents one or more occurrences of the single character (bounded by quotation marks) comprising the literal. An alternative form for "literal" is any figurative constant (not bounded by quotation marks) e.g., ALL SPACES.

Figurative constants generate a string of homogeneous information whose length is determined by the compiler, based upon context. When the length is not deducible from context, a single character is generated. The figurative constants may be used in the Procedure and Data Divisions. When non-numeric figurative constants are used in the Procedure Division, they may not be associated with data items whose length is greater than 120 characters. A figurative constant may be specified in a Record Description VALUE clause for an item longer than 120 characters.

Examples of the above are:

a. MOVE ALL '4' TO COUNT-FIELD, where COUNT-FIELD has been described as having six characters, results in 444444.

b. From the statement MOVE SPACES TO TITLE-BOUNDARY, the compiler will create coding which puts as many space characters into the item TITLE-BOUNDARY as are necessary to fill the item.

c. DISPLAY QUOTE, 'NAME', QUOTE results in 'NAME'.

d. MOVE QUOTE TO AREA-A, where AREA-A has been described as having five characters, results in """".

6. Special Register. TALLY is the name of a special register (USAGE COMPUTATIONAL, SYNCHRONIZED RIGHT) whose length is equivalent to a five decimal digit integer. It may be used to hold intermediate results during the execution of a program.

7. Special Names. Special names provide a means of relating hardware with problem-oriented names, and the status of hardware switches with condition-names.

Verbs

A verb appears in the Procedure Division, and designates an action: ADD, MOVE, GO TO, etc.

Reserved Words

Reserved words are used for syntactical purposes and may not be used as nouns or verbs. There are three types:

1. Connectives. Connectives are words used to:

   a. Denote the presence of a qualifier: OF, IN.

   b. Form compound conditionals: AND, OR, AND NOT, OR NOT; these are called logical connectives.

2. Optional Words. Optional words have been defined and used to improve the readability of the language. Within each format, upper case words which are not underlined are designated as optional. The presence or absence of each optional word within the format for which it is shown does not alter the compiler's translation. However, misspelling of an optional word or its replacement by another word of any kind is not permitted.

3. Key Words. Key words are of three types:

   a. Verbs: ADD, READ, ENTER, etc.

   b. Required words, which appear in formats in various divisions of the language and which are needed to complete the meaning of certain verbs or entries: TO, OMITTED, MEMORY, etc.

   c. Words not shown in any format, but which have a specific functional meaning: NEGATIVE, SECTION, TALLY, etc.

Qualifiers

Every name used in a COBOL source program must be unique, either because no other name has the identical spelling, or because the name exists within a hierarchy of names (so that the name can be made unique by mentioning one or more of the higher levels of the hierarchy). The higher levels are called qualifiers when used in this way, and the process is called qualification. Enough qualification must be mentioned to make the name unique, but it is not necessary to mention all levels of the hierarchy unless they are needed to make the name unique. A file-name is the highest level qualifier available for a data-name. A section-name is the highest and the only qualifier available for a paragraph-name. Thus, file-names and section-names must be unique in themselves and cannot be qualified.

Qualification in COBOL is performed by appending one or more prepositional phrases, using IN or OF. The choice between IN or OF is based on readability, since they are logically equivalent. Nouns must appear in ascending order of hierarchy with either of the words IN or OF separating them. The qualifiers are considered part of the name. Thus, whenever a data item or procedure paragraph is referred to, any necessary qualifiers must be written as part of the name.

7340 Hypertape units may be assigned system names. Specific assignments of this type are made with IBSYS control cards. (See the $FILE card in the publication IBM 7090/7094 Programming Systems: IBJOB Processor, Form C28-6275.) The following restrictions apply:

1. Specification of DENSITY in the RECORDING MODE clause is ignored.
2. If two units have been assigned to a file and the first is a Hypertape unit, the second must also be a Hypertape unit.

e. NONE. When NONE is placed in the Unit 1 field of a $FILE card, that file is said to be absent for that particular run and no input activity will be applied to it.

Within certain limitations, it is permissible to change the contents of the Unit 1 and Unit 2 fields of the $FILE cards after the program has been compiled. One restriction on changing unit assignments after compilation is that files which are assigned to tape media only in the source program (this includes files assigned to NONE in the source program) can not be changed to card media; files which are assigned to card media only in the source program (this includes the printer) can not be changed to tape media. Files which are assigned to either card or tape media in the source program (SYSIN1, SYSOU1, and SYSPP1) may be changed to any other tape or card assignment for a given run of the object program. A file which is assigned to a system unit must conform to the physical and logical characteristics inherent in the system unit. The table on the following pages summarizes the relationships between the permissible forms of the ASSIGN TO clause and the contents of associated fields in the $FILE cards which are produced by the compiler.

I-O-CONTROL

Function

To specify rerun procedures.

```
I-O-CONTROL.

RERUN [ON CHECKPOINT-UNIT]      EVERY

BEGINNING OF REEL OF file-name-1

  [, file-name-2. . .]  [RERUN. . .] .
```

Notes:

1. This paragraph is required only when rerun is desired.
2. RERUN may be used to specify that memory dumps are to be written under one of the following conditions:

   a. At each reel switch of the specified file (whether input or output), a memory dump is to be written on the unit which is provided by the compiler as the standard checkpoint-unit.

File Assignment Table

Form of ASSIGN TO Clause	Contents of Associated $FILE Card Fields		
	Unit 1 Field	Unit 2 Field	Multireel Field
1 TAPE-UNIT	omitted	omitted	omitted
1 TAPE-UNIT MULTIPLE REEL	omitted	omitted	REELS
2 TAPE-UNITS MULTIPLE REEL	*	omitted	REELS
CARD-READER	CRD	omitted	omitted
CARD-PUNCH	PCH	omitted	omitted
PRINTER	PRT	omitted	omitted
symbolic-tape-unit-name-1	symbolic-tape-unit-name-1	omitted	omitted
Example: A	A	omitted	omitted
symbolic-tape-unit-name-2 MULTIPLE REEL	symbolic-tape-unit-name-2	omitted	REELS
Example: TIV MULTIPLE REEL	TIV	omitted	REELS
symbolic-tape-unit-name-3 symbolic-tape-unit-name-4 FOR MULTIPLE REEL	symbolic-tape-unit-name-3	symbolic-tape-unit-name-4	REELS
Example: C(3), C(4) MULTIPLE REEL	C(3)	C(4)	REELS
symbolic-card-unit-name-1	symbolic-card-unit-name-1	omitted	omitted
Example: RDA	RDA	omitted	omitted
system-unit-name-1	abbreviated system-unit-name-1	omitted or may be filled in automatically	
Example: SYSIN1 #	IN1	IN2	REELS
system-unit-name-2 MULTIPLE REEL	abbreviated system-unit-name-2	omitted or may be filled in automatically	REELS
Example: SYSUT1 MULTIPLE REEL	UT1	omitted	REELS
system-unit-name-3, system-unit-name-4	abbreviated system-unit-name-3	abbreviated system-unit-name-4	REELS
Example: SYSUT2, SYSUT3 FOR MULTIPLE REEL	UT2	UT3	REELS
NONE	NONE	omitted	omitted

#Note: In the example, this particular file automatically takes on the MULTIREEL characteristic of the system unit. This is also true for SYSOU1 and SYSPP1 which are system MULTIREEL files.

2. There is only one storage area reserved for all standard labels, whether input or output, and all labels are processed in that area. The VALUE clause, described under VALUE, provides the only means of referring to this area.

3. Standard labels are of two types, header labels and trailer labels. The various items within a label are processed in different ways depending upon which type label is being read.

4. Since there is only one label area shared by all files, any references to items within that area must not be qualified by a file name, i.e., each name in the label area is unique.

5. The standard label area is processed by the IOCS for any one of the following purposes:

   a. To check an input file header label. A header label occurs at the beginning of the file and at the beginning of every reel of the file. At the time this type of label is processed by IOCS, the items FILE-SERIAL-NUMBER, REEL-SEQUENCE-NUMBER, and FILE-IDENTIFICATION are checked for the values designated in the respective VALUE OF clauses under the FD entry for the file.

   b. To check an input file trailer label. A trailer label occurs at the end of the file and at the end of every reel of the file. If the value of the item LABEL-IDENTIFIER is '1EORb', the label is an END-OF-REEL label; if the value is '1EOFb', the label is an END-OF-FILE label. At this time the BLOCK-COUNT item (this item applies to trailer labels only) is checked against the number of blocks actually read by the IOCS.

   c. To check the header label on a tape to be used in order to insure that the tape may be written upon. Tapes that have labels beginning with '1BLANK' are accepted unconditionally, but those tapes beginning with '1HDRbb' are accepted only if RETENTION-PERIOD indicates that the life of the tape has expired.

   d. To prepare an output file header label. At the time this label is formed by the IOCS, the items RETENTION-PERIOD, REEL-SEQUENCE-NUMBER, FILE-IDENTIFICATION and FILE-SERIAL-NUMBER are formed from the information supplied in the respective VALUE OF clauses under the FD entry for the file.

      If these fields have not been specified by $LABEL cards and if the VALUE clauses have been omitted, IOCS supplies the following:

RETENTION-PERIOD:	000	(days)
REEL-SEQUENCE-NUMBER:	b0001b	(for a file being opened and consecutively higher for succeeding reels)
FILE-SERIAL-NUMBER:		the file-serial-number from the label of the tape being written over.
FILE-IDENTIFICATION:		the file-name given by the source programmer, which may have been modified by the compiler to limit the name to 18 characters.

e. To prepare an output file trailer label. At this time the LABEL-IDENTIFIER item is filled with the proper END-OF-REEL or END-OF-FILE indication, and the BLOCK-COUNT item is also filled in with the actual number of physical blocks that have been written on the reel.

6. Files that are assigned to SYSIN1, SYSOU1, or SYSPP1 may not be labeled.

RECORD SIZE

Function

To specify the size of data records.

```
[, RECORD CONTAINS  [integer-3 TO]  integer-4  CHARACTERS]
```

Notes:

1. Integer-3 and integer-4 must be numeric literals with positive integral values.
2. The size of each data record is completely defined within the Record Description entries; therefore this clause is never required. When present, however, the following notes apply:
   a. In this clause, CHARACTERS refers to the number of computer characters which the record will occupy. Therefore, when specifying the number of computer characters, careful consideration must be given to items which are either synchronized or computational.
   b. Integer-4 may not be used by itself unless all the data records in the file have the same size. In this case integer-4 represents the exact number of characters in the data record. If integer-3 and integer-4 are both shown, they refer to the minimum number of characters in the smallest size of data record and the maximum number of characters in the largest size of data record, respectively.

RECORDING MODE

Function

To specify the external format of a file.

```
[, RECORDING MODE IS  {BCD}  [{LOW}  DENSITY]
                      {BINARY} [{HIGH}
[ WITHOUT COUNT CONTROL ]]
```

Notes:

1. The BCD mode should be specified for any file assigned exclusively to card equipment. If the unit assigned to the file may be either a tape or card unit, as is the case with some system units, the recording mode must agree with the system unit recording mode.
2. If information is recorded on the magnetic tape just as it is found in core storage (except for check bits), the tape is said to be a binary tape or to have been written in the binary recording mode.

Through the use of auxiliary units, alphanumeric information may be recorded on or read from a magnetic tape independently of the computer. Magnetic tapes prepared or used by this auxiliary equipment employ a special coding system known as binary-coded-decimal. Tapes which are prepared using the binary-coded-decimal system are said to be BCD tapes, i.e., the RECORDING MODE IS BCD. Tapes recorded in the BCD mode cannot contain data items whose USAGE IS COMPUTATIONAL. Therefore, all data items recorded in the BCD mode must be described as USAGE IS DISPLAY.

3. Each tape unit is capable of writing characters on magnetic tapes at two densities, LOW or HIGH. The density of a tape refers to the number of characters that can be written on a given area of tape. The more characters that can be written, the higher the character density. Some auxiliary equipment requires information to be recorded in LOW DENSITY. However, if the tape is to be completely processed on the 7090/94, the HIGH DENSITY option should be used to minimize the tape area required to contain a given number of characters.

4. If the clause is not used, RECORDING MODE will be BCD, HIGH DENSITY.

5. When the WITHOUT COUNT CONTROL option is specified, control words preceding logical records on files containing logical records of different sizes will not be assumed when reading and will not be supplied when writing.

   A discussion of buffer size calculations is contained in the discussion of BLOCK SIZE.

VALUE

Function

To specify the contents of particular items in the standard labels associated with a file.

[, VALUE OF label-data-name-1 IS literal-1 [label-data-name-2 IS ...]]

Notes:

1. Each label-data-name is a data-name item and must be an item in the standard label area (see LABEL RECORDS).

2. When label-data-name is alphanumeric, literal-1 must be a non-numeric literal (enclosed within quotation marks). When label-data-name is numeric, literal-1 must not be enclosed in quotation marks, and must consist only of numerals.

3. Files may have label-data-names conforming to the following lists:

a. Input files.

Label-data-name	Contents of literal
FILE-IDENTIFICATION	Eighteen or less alpha-numeric characters which identify the file.
FILE-SERIAL-NUMBER	Five or less alphabetic and/or numeric characters (no special characters) with no embedded blanks. This is the REEL-SERIAL-NUMBER of the file. The REEL-SERIAL-NUMBER of a reel of tape is the number on the external casing of that reel.
REEL-SEQUENCE-NUMBER	Four or less numerals. This is the number of the reel within a given file, i.e., the first reel of a file is reel 1, the second is reel 2. The value specified in the VALUE OF clause is checked against the value in the label of the first reel of the file which is processed. Every time a reel switch occurs the value is increased by one, and the new value is checked against the value in the label of the new reel.

Each of the above items may or may not be specified for a standard label input file. If the item is not specified, it is not checked when the file is processed.

b. Output files.

Label-data-name	Contents of literal
FILE-IDENTIFICATION	Eighteen or less alpha-numeric characters which identify the file.
RETENTION-PERIOD	Four or less numerals. This is the number of days that the file is to be saved after its date of creation.

Label-data-name	Contents of Literal
	The creation date for the file is supplied by the input/output system at the time the file is created.
FILE-SERIAL-NUMBER	Five or less alphabetic and/or numeric characters (no special characters) with no embedded blanks. This item has the same definition as input files. (See above.)
REEL-SEQUENCE-NUMBER	Four or less numerals. This item has the same definition as above.

Each of the preceding items may or may not be specified for a standard label output file. The IOCS applies the following procedures to these items:

1. If FILE-IDENTIFICATION is specified, the non-numeric literal is placed in the labels created. If it is not specified, the first eighteen characters of the file name are placed in the labels created.

2. If RETENTION-PERIOD is specified, that value is placed in the labels created. If it is not specified, the value placed in the labels created will be zero.

3. The value of FILE-SERIAL-NUMBER is placed in the label only if the REEL-SEQUENCE-NUMBER value specified is greater than one. If the REEL-SEQUENCE-NUMBER is not specified as greater than one, the value used for FILE-SERIAL-NUMBER in creating a new label will come from the label of the reel chosen to be the next one used.

4. If REEL-SEQUENCE-NUMBER is specified, that value will be placed in the label of the first reel of the file. Each reel after the first reel will get a REEL-SEQUENCE-NUMBER one greater than the previous reel. If REEL-SEQUENCE-NUMBER is not specified, the value used for the first reel will be one.

RECORD DESCRIPTION

General Description

Elements of a Detailed Data Description

A Detailed Data Description consists of a set of entries. Each entry defines the characteristics of a particular unit of data. With minor exceptions, each entry is capable of completely defining a unit of data. Because the COBOL Detailed Data Descriptions involve a hierarchical structure, the contents of an entry may vary considerably, depending upon whether or not it is followed by subordinate entries.

In defining the lowest level or subdivision of data, the following information may be required:

a. A level-number which shows the relationship between this and other units of data.

b. A data-name.

c. The size in terms of the number of digits or characters.

d. The usage of the data.

e. The number of consecutive occurrences (OCCURS) of elements in a table or list.

f. The class or type of data, (ALPHABETIC, NUMERIC, or ALPHANUMERIC).

g. The type of sign.

h. Location of an actual or an assumed decimal point.

i. Location of editing symbols such as dollar signs and commas.

j. Synchronization of the data (SYNCHRONIZED).

k. Special editing requirements such as zero suppression and check protection.

l. Initial value (VALUE) of a working-storage item or the fixed value (VALUE) of a constant.

An entry which defines a unit of data must not be contradicted by a subordinate entry. Thus, once the class is defined, it applies to all subordinate entries and need not be re-specified. However, when the class is defined as alphanumeric, subordinate entries may particularize the class by specifying alphabetic or numeric. If the class has been defined as either alphabetic or numeric, subordinate entries may not change the class.

PICTURE

Function

To show a detailed picture of an elementary item, the general characteristics of the item, and special report editing.

[, PICTURE IS (any allowable combination of characters and symbols as described below)]

Notes:

Because the choice of characters in any given PICTURE depends on the type of data item being described, the characters will be grouped according to the type of data item they describe.

1. Numeric Items

   The PICTURE of any numeric data item may contain a combination of only the following characters:

   9 V P S

   The significance of these four characters is as follows:

   a. The character 9 indicates that the character position will always contain a numeric character.

   b. The symbol V indicates the position of an assumed decimal point. Since a numeric item cannot contain an actual decimal point, an assumed decimal point is used to provide the compiler with the information concerning the alignment of items involved in computation.

   c. The character P indicates an assumed decimal scaling position. It is used to specify the location of an assumed decimal point when the point is not within the data item. The character V may be used or omitted as desired. If it is used, however, it must be placed in the position of the assumed decimal point.

   d. The character S is equivalent to the SIGNED clause and indicates the presence of an operational sign. If used, it must always be written as the leftmost character of the PICTURE. If the USAGE of the item is COMPUTATIONAL, the character S in a PICTURE is a redundant specification. If the USAGE of the item is DISPLAY, it specifies a sign overpunch in the units position.

2. Alphabetic Items

The PICTURE of an alphabetic item can contain only the character A.

An A indicates that the character position will always contain an alphabetic character, i.e., a letter or a space.

3. Alphanumeric Items

An alphanumeric item has been defined as an item which may contain any character in the character set of the computer. Alphanumeric items can be divided into two types: non-report items for which editing is not specified, and report items for which editing has been specified.

a. Non-report Items

The PICTURE of a non-report item may contain only the characters 9, A, and X and must contain at least one A or X. The characters 9 and A have been discussed above. A mixture is treated as if all the characters were Xs.

An X indicates that the character position may contain any character in the computer's character set.

b. Report Items

Editing of numeric data is accomplished by moving the data to a report item which specifies the insertion, replacement, and/or suppression of certain characters. The PICTURE of a report item may contain a combination of the following characters:

9 V P , . + - Z * CR DB B 0 $

The uses of 9, V, and P have been discussed above. The remaining characters will be explained in three groups: zero suppression, insertion, and replacement characters.

Zero suppression or character replacement is accomplished by placing a character designated for the desired editing in each leading numeric character position that is to be suppressed or replaced. Three general rules apply:

(1) Suppression and/or replacement terminates with the character immediately preceding the first digit other than zero, or the decimal point (assumed or actual) whichever is encountered first.

(2) If all numeric character positions in a PICTURE reserved for source data (as opposed to those additional positions used for insertion characters) contain suppression characters (asterisk is excluded), then all characters will be replaced by spaces if the value of the source data is zero.

The insertion characters B and O are treated like the insertion character comma in these two ways:

1. If all characters to the left of the insertion character have been suppressed, the insertion character will be suppressed.

2. Multiple contiguous insertion characters are not allowed. If a string of identical characters is specified, it will be replaced by a single insertion character and a warning message will be issued. Contiguous insertion characters of different types will cause an E level message to be issued.

Replacement Characters. The asterisk indicates check protection, i.e., the replacement of non-significant zeros by asterisks.

The Floating Dollar Sign. Zero suppression with a floating dollar sign is specified by placing a dollar sign in each numeric character position to be suppressed. A dollar sign will be placed in the rightmost position in which suppression is to occur. All floating dollar signs must be the leftmost characters in a PICTURE with the exception that single commas, B's, or 0's may be embedded. Suppression of leading commas, B's, and 0's is also provided.

The Floating Minus Sign. Zero suppression with a floating minus sign is specified by placing a minus sign in each numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur. If the value is positive or zero, a space will be inserted instead of a minus sign. All floating

(3) In any picture containing a string of floating characters, the number of characters reserved by the picture must reflect the existence of this editing character. Thus, the picture of a string of floating characters must be at least one character larger than the greatest number that will be moved into it.

Zero Suppression Character. The character Z specifies that the character position is to be suppressed by a space if a non-significant zero appears in the position. Z must never be preceded by 9.

Insertion Characters. The single dollar sign character specifies that a $ is to be placed in the indicated position. It must be the first character in the PICTURE.

The single minus sign, written either as the first or the last character of a PICTURE, specifies that a display minus sign is to be placed in the indicated position when the value of the source item is negative. If the value is not negative, a space will be inserted.

The single plus sign, written either as the first or the last character of a PICTURE, specifies that a display minus sign is to be placed in the indicated position if the value of the source item is negative. If the value of the item is not negative, a display plus sign will be inserted.

The comma character specifies that a comma is to be inserted in the indicated position unless the following condition occurs: If the suppression or replacement has caused the elimination of all digits to the left of the comma, the comma itself will be suppressed.

The decimal point character specifies that an actual decimal point is to be inserted in the indicated position and the source item is to be aligned accordingly. Numeric character positions to the right of an actual decimal point in a PICTURE must consist of characters of one type.

CR and DB are called the "credit" and "debit" symbols and may appear only at the right end of a PICTURE. These symbols occupy two character positions and indicate that the specified symbol is to appear in the indicated positions if the value of a source item is negative. If the value is positive or zero, spaces will appear instead.

The zero specifies that the character 0 is to be inserted in the indicated position.

The character B specifies that a space is to be inserted in the indicated position.

minus signs must be the leftmost characters in a PICTURE with the exception that single commas, B's, or 0's may be embedded. Suppression of leading commas, B's, and 0's is also provided.

The Floating Plus Sign. Zero suppression by means of a floating plus sign is specified by placing a plus sign in each leading numeric character position to be suppressed. If the value of the item is negative, a minus sign will be placed in the rightmost position in which suppression is to occur. If the value of the item is not negative, a plus sign will be inserted instead. All floating plus signs must be leftmost characters in a PICTURE except that single commas, B's, or 0's may be embedded. Suppression of leading commas, B's, and 0's is also provided.

c. Scientific Decimal Items.

A scientific decimal item is a special type of report item which specifies editing of a floating-point number. The PICTURE of a scientific decimal item may contain only the following characters:

+ - 9 . V E

Specifically the PICTURE must conform to the form:

$$\underbrace{\{\pm\}\ [9(m)]\left[\begin{Bmatrix}V\\.\end{Bmatrix}\right][9(n)]\ E}_{\text{Mantissa}}\ \underbrace{\{\pm\}\ 99}_{\text{Exponent}}$$

where m and n are positive integers and m+n must be greater than 0 and less than 17. The symbol E is used to give visual separation of the exponent from the mantissa.

4. Record Mark.

The PICTURE character J specifies that the single character called record mark (of special importance to some peripheral equipment) is to appear in the indicated position as a constant. If J is used, the PICTURE must contain only one J, and no other character may appear with it.

The data item being specified is one alphanumeric character whose value is a record mark (‡) and all the rules for a data item with such specifications apply

5. General Notes.

a. A PICTURE clause can be used only at the elementary item level.

b. An integer which is enclosed in parentheses following any symbol listed below indicates the number of consecutive occurrences of that symbol:

A X 9 P Z * $ B 0 - +

c. The maximum number of characters allowed in a PICTURE is 30.

b. Comparison of Non-numeric Items. For two non-numeric items, a comparison results in the determination that one of the items is less than, equal to, or greater than the other with respect to a collating sequence. The standard collating sequence is the commercial collating sequence, but an alternate sequence may be specified by designating that the 7090/7094 collating sequence is to be used. This is specified by placing the option BINSEQ on the $IBCBC control card.

The figurative constant HIGH-VALUE [S] assumes the value of left parenthesis in the 7090/94 collating sequence and 9 in the commercial collating sequence. The figurative constant LOW-VALUES [S] assumes the value of zero in the 7090/94 collating sequence and blank in the commercial collating sequence.

COLLATING SEQUENCE

The two permissible collating sequences, in order, with the lowest values at the top of the columns, are as follows:

7090/94	commercial
0 through 9	blank or space
=	.
'	) or ¤
+	+ or &
A through I	$
± 0	*
.	-
)	/
-	,
J through R	(or %
0̄	= or #
$	' or @
*	± 0
blank or space	A through I
/	0̄
S through Z	J through R
≠	≠
,	S through Z
(	0 through 9

The commercial collating sequence is assumed by the Compiler unless the 7090/7094 collating sequence is specified by placing BINSEQ on the $IBCBC card.

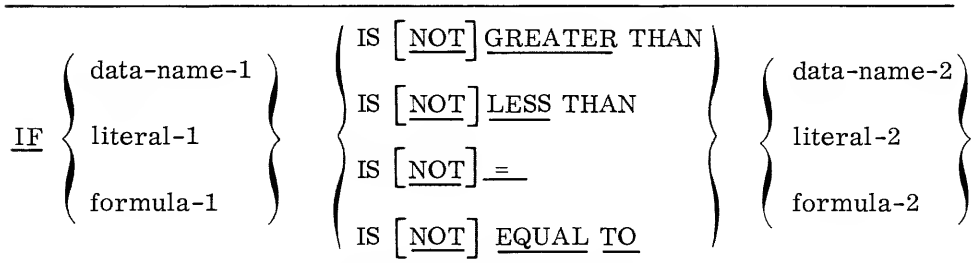
2. Full Relation Test Format. There are two cases to consider: equal length items, and unequal length items.

   a. Items of Equal Length. If the items are of equal length, comparison proceeds by comparing characters in corresponding character positions starting from the high order end and continuing until either a pair of unequal characters is encountered or the low order end of the item is reached, whichever comes first. The items are determined to be equal when the low order end is reached.

      The first pair of unequal characters encountered is compared for relative location in the ordered character set. The item which contains that character which is positioned higher in the ordered sequence is determined to be the greater item.

   b. Items of Unequal Length. If the items are of unequal length, comparison proceeds as described above. If this process exhausts the characters of the shorter item, then the shorter item is less than the longer item unless the remainder of the longer item consists solely of spaces, in which case the two items are equal.

The format for full relation tests is:



In the above format, the actual choice from data-name-1, literal-1, or formula-1 is called the subject. The choice from data-name-2, literal-2, or formula-2 is called the object. The subject and the object cannot both be literals.

the normal hierarchical sequence of execution in formulas where it is necessary to have some deviation from the normal precedence. When the sequence of execution is not specified by parentheses, the order of execution of consecutive operations of the same hierarchical level is from left to right. Thus, expressions ordinarily considered to be ambiguous, e.g., A/B*C and A/B/C are permitted in COBOL. They are interpreted as if they were written (A/B) * C, and (A/B) / C, respectively. Without parenthesizing, the following example illustrates normal precedence:

```
A + B / C + D ** E * F - G
```

would be interpreted as if written:

```
A + (B / C) + ((D ** E) * F) - G
```

with the sequence of operations working from the inner-most parentheses toward the outside. Exponentiation will be performed first, then multiplication and division and finally addition and subtraction. Exponentiation of a negative variable or literal is allowed only if the exponent is a literal or data-name having an integral value. If the exponent is other than a non-negative integer, the result of the exponentiation will be zero.

Formation Of Symbol Pairs

The ways in which symbol pairs may be formed are summarized in the table below, where P indicates a permissible pair.

		SECOND SYMBOL				
		VARIABLE	* / **	-+	(	)
	VARIABLE	-	P	P	-	P
FIRST SYMBOL	* / **	P	-	P	P	-
	+ -	P	-	-	P	-
	(	P	-	P	P	-
	)	-	P	P	-	P

VERBS

Listed By Categories

Arithmetic

- ADD
- SUBTRACT
- MULTIPLY
- DIVIDE
- COMPUTE

Input-Output	READ WRITE OPEN CLOSE DISPLAY ACCEPT
Procedure Control	GO TO ALTER PERFORM
Data Movement	EXAMINE MOVE
Ending	STOP
Compiler Directing Verbs	ENTER EXIT NOTE USE

Note: Although the word IF is not a verb in the strictest sense, it possesses one of the most important characteristics of one, namely the generation of coding in the object program. Its occurrence is a vital feature in the Procedure Division.

Specific Verb Formats

The specific verb formats, together with a detailed discussion of the restrictions and limitations associated with each, appear on the following pages, in alphabetic sequence.

ACCEPT

Function

To make available low-volume input data from the card reader or from SYSIN1, the system input unit.

ACCEPT data-name [FROM SYSIN1]

Notes:

1. The length of the item may not exceed 72 characters.
2. Data-name must be USAGE DISPLAY and may only be an alphanumeric or alphabetic non-report item, an external decimal item, or a group item.
3. No editing or error checking of the value of data name is performed.

ADD

Function

To add two or more numeric data items and set the value of an item equal to the result.

Option 1:

```
ADD { literal-1   } [ , { literal-2   } ... ] [ { TO     } data-name-n ]
    { data-name-1 }     { data-name-2 }         { GIVING }

[ ROUNDED ] [ , ON SIZE ERROR { imperative-statement-1 } ]
                              { NEXT SENTENCE          }

[ { ELSE      } { imperative-statement-2 } ]
  { OTHERWISE }   { NEXT SENTENCE        }
```

Option 2:

```
ADD CORRESPONDING data-name-1 TO data-name-2

[ ROUNDED ] [ , ON SIZE ERROR { imperative-statement-1 } ]
                              { NEXT SENTENCE          }

[ { ELSE      } { imperative-statement-2 } ]
  { OTHERWISE }   { NEXT SENTENCE        }
```

Notes:

1. Except when the CORRESPONDING option is used, the data-names used must refer only to the special register, TALLY, or to numeric elementary items whose descriptions appear in the Data Division of the program.

2. The maximum size of any operand (literal or data-name) is 18 decimal digits. An error will be indicated at compilation time if the format for any operand specifies a number of digits in excess of 18. Intermediate results are carried to a maximum of 20 digits with no loss of least significant digits except when the maximum is reached.

3. If the GIVING option is used, the value of data-name-n will be made equal to the sum of the values of the preceding data-names and/or literals. Data-name-n is not used as an addend in this option; hence its format may contain editing symbols.

If the TO option is used, the sum of the values of data-name-n and the preceding data-names and/or literals will be calculated; the value of data-name-n will then be made equal to the sum.

If neither the GIVING nor the TO options are used, the last named (the rightmost or last written) addend must not be a literal.

The sum of the values of all the data-names and/or literals will be calculated. The value of the rightmost data-name will then be made equal to the sum. Since it is used as an addend in this case the format of the rightmost data-name may not contain any editing symbols.

Examples:

Statement	Sum	Stored In
ADD A, B, C	A+B+C	C
ADD A, B, TO C	A+B+C	C
ADD A, B, C GIVING D	A+B+C	D

4. An error will be indicated at compilation time if the data description of any item used as an addend specifies the presence of editing symbols. Operational signs and implied decimal points are not considered editing symbols. Literals used as addends must be numeric.

5. An ADD statement must refer to at least two addends.

6. The formats associated with all operands referred to in an ADD statement may differ among each other. Decimal point alignment is automatically supplied throughout the calculation.

7. If the number of decimal places in the calculated result (sum) is greater than the number of decimal places associated with the resultant data-name (the data-name whose value is to be set equal to the sum), truncation will occur unless the ROUNDED option has been specified. Truncation is always in accordance with the size associated with the resultant data-name. When the ROUNDED option is specified, the least significant digit of the resultant data-name has its value increased by 1 whenever the most significant digit of the excess is greater than or equal to 5.

8. Whenever the number of integral places (those to the left of the decimal point) in the calculated result exceeds the number of integral places associated with the resultant data-name, a size error condition arises.

   In the event of a size error condition, one of two possibilities will occur, depending on whether or not the ON SIZE ERROR option has been specified.

a. The testing for the size error condition occurs only when the ON SIZE ERROR option is specified in the verb format. In the event that ON SIZE ERROR is not specified, and a size error condition arises, the effect will be unpredictable.

b. If the ON SIZE ERROR option has been specified, and a size error condition arises, the value of the resultant data-name will be altered unpredictably, and the imperative-statement-1 associated with the ON SIZE ERROR option will be executed.

9. When the CORRESPONDING option is used, the treatment of the data names is similar to that in the CORRESPONDING option of the MOVE statement except that the quantities are added rather than moved. See Notes 6 through 10 in the section on the MOVE verb.

ALTER

Function

To modify a predetermined sequence of operations.

```
ALTER   procedure-name-1   TO PROCEED TO   procedure-name-2
      [ procedure-name-3   TO PROCEED TO   procedure-name-4 ... ]
```

Notes:

1. Procedure-name-1, procedure-name-3, ..., are names of paragraphs, each containing a single sentence consisting of only a GO TO statement as defined under option 1 of the GO TO verb.

2. The effect of an ALTER statement is to replace the procedure-name specified in the GO TO sentence (located at procedure-name-1) by the procedure-name-2 specified in the ALTER statement.

CLOSE

Function

To terminate the processing of input and output reels and files, with optional rewind and/or lock.

```
CLOSE   file-name-1 [REEL] [WITH { NO REWIND } ][, file-name-2 ...]
                                 { LOCK      }
```

Notes:

1. The CLOSE file-name option (as applied to the entire file rather than to individual reels) will initiate the final closing conventions for the file and release the data area. The CLOSE verb may be applied to any file with OPEN status but must not be re-used on the same file without an intervening OPEN instruction.

2. CLOSE file-name (without the REEL option) will have the following effects:

   a. Input Files.

      (1) If neither NO REWIND nor LOCK is specified, the current reel of the file will be rewound.

      (2) If the NO REWIND option is specified, the current reel of the file will not be rewound.

      (3) If the LOCK option is specified, the current reel of the file will be rewound and unloaded.

   Any label processing and procedures specified in a USE Declarative are performed only when the physical end of file is encountered on the tape.

   b. Output Files.
   The final closing conventions and procedures specified in a USE Declarative are performed and the data area is released.

      (1) If neither LOCK nor NO REWIND is specified, the current reel of the file will be rewound.

      (2) If the NO REWIND option is used on a tape file, the last reel of the file will remain positioned at the end of the file.

      (3) The use of the LOCK option will rewind and unload the current reel.

3. If the CLOSE file-name REEL option is used, then, for both input and output files, the next reel processing procedures are instituted. More specifically:

   a. Input Files.

      Processing of the end label will be bypassed, but procedures for checking the label on the next reel, including procedures specified in a USE Declarative, will be executed. If a CLOSE REEL is given for the last reel of a file, an error will occur in the object program. Furthermore,

Notes:

1. When using option 1, if the GO TO statement is to be modified by the ALTER verb:
   a. The GO TO statement must itself have a paragraph-name.
   b. The paragraph in which the GO TO statement is included must consist solely of the GO TO statement.

   The paragraph-name assigned to the GO TO statement is referred to by using ALTER verb in order to modify the sequence of the program. If procedure-name-1 is omitted, and if the GO TO statement is not referred to by an ALTER statement prior to the first execution of the GO TO statement, execution of the program will be terminated and control will be returned to IBJOB.

2. In option 2, the contents of data-name must have a positive integral value at object time. The branch will be to the 1st, 2nd, ..., nth procedure-name, as the value of data-name is 1, 2, ..., n. If the value of data-name is anything other than the integers 1, 2, ..., n, then no transfer is executed and control passes to the next statement in the normal sequence for execution.

MOVE

Function

To transfer data, in accordance with the rules of editing, to one or more data areas.

Option 1:

MOVE {data-name-1 / literal} TO data-name-2 [, data-name-3...]

Option 2:

MOVE CORRESPONDING data-name-1 TO

data-name-2 [, data-name-3...]

Notes:

1. Additional receiving areas may be given, following data-name-2. The data designated by the literal or data-name-1 will be moved first to data-name-2, then to data-name-3, etc. When data-name-2 is referred to in this discussion, the note also applies to the other receiving areas.

2. It is improper to use MOVE (without the CORRESPONDING option) for a group item whose format is such that editing would be required on the elementary items in separate operations. If this type of procedure is desired, the CORRESPONDING option must be used, or else each elementary item must be handled individually by means of the verb, MOVE.

3. At object time, data is stored in conformity with the description of the receiving area. When the sizes of the areas of two group items involved in a move are not the same, a warning will be given by the compiler during compilation. A warning is also given for a move from an elementary to a group item or vice versa.

4. When numeric elementary items are moved, they are subject to the following procedures:

   a. They are aligned by decimal points, with zero filling or truncation on either end as required. A warning is given by the compiler if significant digits will be lost through truncation.

   b. They may be converted from one form to another, e.g., internal decimal to external decimal, numeric mode to alphanumeric mode.

   c. They may have special editing performed on them with suppression of zeros, insertion of dollar sign, commas, a decimal point, etc., and alignment as specified by the description of the receiving area. The presence of these special characters in an item actually makes the item alphanumeric. If such an item is referred to a source, the special characters will be picked up as a part of the data, and the verb making the reference will treat this data according to the rules specified for the treatment of alphanumeric data.

5. When non-numeric items are moved:

   a. The characters are placed in the receiving area left to right.

   b. If the source field is shorter than the receiving field, the remaining character positions are filled with spaces.

   c. If the source field is longer than the receiving field, the generated code provides for termination of the move as soon as the receiving field has been filled. A warning message is given for this condition unless one has already been given as stated in note 3.

   d. A table of legal moves, using the verb MOVE, is given below.

      A detailed description of the types of fields represented may be found under the PICTURE clause in the Data Division. Numbers in parentheses in the table refer to subsequent notes.

Source Field Type	Receiving Field Type: Group Item (1)	Alpha-betic	Alpha-numeric	Report	External Decimal	Internal Decimal	Floating Point	Scientific Decimal
Group Item (1)	yes	yes	yes	no	no	no	no	no
Alphabetic	yes	yes	yes	no	no	no	no	no
Alphanumeric	yes	yes	yes	no	no	no	no	no
Report	yes	no	yes	no	no	no	no	no
External Decimal	yes	no	yes	yes	yes	yes	yes	yes
Internal Decimal	yes	no	yes	yes	yes	yes	yes	yes
Floating Point	yes	no	yes	yes	yes	yes	yes	yes
Scientific Decimal	yes	no	yes	yes	yes	yes	yes	yes
ZERO [S] or ZEROES	yes	no	yes	yes(2)	yes	yes	yes	yes(2)
SPACE [S]	yes	yes	yes	yes(3)	nc	no	no	no
LOW-VALUE [S]	yes	no	yes	no	no	no	no	no
HIGH-VALUE [S]	yes	no	yes	no	no	no	no	no
QUOTE [S]	yes	no	yes	no	no	no	no	no
ALL---	yes	yes(4)	yes	no	yes(5)	no	no	no

(1) Group items are treated as having a PICTURE of all X's.

(2) The value zero is moved in accordance with editing requirements.

(3) A warning message is given.

(4) The character must be alphabetic.

(5) The character must be numeric.

e. For source fields of the scientific decimal type:

At object time a free form of data is allowed within the limits of the field. For example, a field with PICTURE -99V9E-99 may contain the value 1 in any of the following ways (where b represents a space):

b . 01b2b	(01×10^2)
1bb+01b	(note scale applied when no point)
.001bb3	
b10bbbb	(note scale applied when no point)
1000.-3	
etc.	

Note that the letter E is never part of the data.

6. If the CORRESPONDING option is used, selected items within data-name-1 are moved, with any required editing, to selected areas within data-name-2. Items are selected by matching the data-names of areas defined within data-name-1 with like data-names of areas defined within data-name-2, according to the following rules:

   a. At least one of the items of a selected pair must be an elementary item.

   b. The respective data-names are the same including all qualification up to but not including data-name-1 and data-name-2.

   Each CORRESPONDING source item is moved in conformity with the description of the receiving area. The results are the same as if the user had referred to each pair of CORRESPONDING data-names in separate MOVE statements.

7. When using a MOVE CORRESPONDING, only the first complete description of any area will be considered in the case where a REDEFINE has been used. All items describing the data contained within a REDEFINE group will be ignored.

8. If the CORRESPONDING option is used, no items in the group referred to can contain an OCCURS clause.

Notes:

1. An OPEN statement for the file must be executed prior to the execution of the first READ for that file.

2. When a file consists of more than one type of logical record, these records automatically share the same storage area. This is equivalent to saying that there exists an implicit redefinition of the area, and only the information which is present in the current record is accessible.

3. No reference can be made by any verb in the Procedure Division to information which is not actually present in the current record. Thus, it is not allowable to refer to the nth occurence of data which appears fewer than n times. If such a reference is made the results in the object program are unpredictable.

4. When the INTO data-name option is used, the data-name must be the name of a Working-Storage or output record area. If the format of the data-name differs from that of the input record, moving will be performed according to the rules specified for the MOVE verb without the CORRESPONDING option.

   When the INTO data-name option is used, the file-name RECORD is still available in the input record area.

5. Every READ statement must have an AT END clause.

6. After execution of the AT END clause, an attempt to perform a READ without the execution of a CLOSE and a subsequent OPEN for that file will constitute an error in the object program except for multi-reel unlabeled tape files.

   For multi-reel unlabeled files the programmer identifies the file in the File-Control paragraph as a multiple reel file and provides a means of communicating to the program, usually by sense switches or control cards, whether a continuation reel exists for the file. Upon execution of the AT END clause, if it is determined that the last reel of the file has been processed, the CLOSE clause may be executed. If it is determined that a continuation reel does exist for the file, the next READ statement for the file will cause IOCS to rewind and unload the reel last read and to effect unit switching if required.

7. After recognition of the end of reel, the READ performs the following operations:

   a. The standard trailer label subroutine and procedures specified by a USE Declarative.

   b. A tape alternation.

   c. The standard header label subroutine and procedures specified by a USE Declarative.

   d. Makes the next record available.

STOP

Function

To halt the object program either permanently or temporarily.

```
STOP { literal }
     { RUN   }
```

Notes:

1. If the word RUN is used, then the ending procedure established by the compiler is instituted.

2. If the literal form is used, the literal will be displayed to the operator on the on-line printer.

   Continuation of the object program will begin with the execution of the next statement in sequence.

SUBTRACT

Function

To subtract one, or a sum of two or more, numeric data items from a specified item and set the value of an item equal to the result.

```
Option 1:
SUBTRACT { literal-1 } [ , { literal-2 } ... ]
         { data-name-1 }   { data-name-2 }

FROM { literal-n } [ GIVING data-name-m ] [ ROUNDED ]
     { data-name-n }

[ , ON SIZE ERROR { imperative-statement-1 } ]
                  { NEXT SENTENCE          }

[ { ELSE      } { imperative-statement-2 } ]
  { OTHERWISE } { NEXT SENTENCE          }
```

Option 2:

SUBTRACT CORRESPONDING data-name-1 FROM

data-name-2 [ROUNDED] [, ON SIZE ERROR

{imperative-statement-1 / NEXT SENTENCE} [{ELSE / OTHERWISE}

{imperative-statement-2 / NEXT SENTENCE}]]

Notes:

1. Except when the CORRESPONDING option is used, all data-names used must refer to the special register, TALLY, or to numeric elementary items whose descriptions appear in the Data Division of the source program.

2. All rules specified under the verb ADD with respect to the ON SIZE ERROR option, the size of operands, the ROUNDED option, the GIVING option, truncation, and the editing of results, apply to the SUBTRACT verb. (See ADD.)

3. If the data description of any literal or data-name used as either the minuend or the subtrahend specifies the presence of an editing symbol, an error will be indicated at compilation time. Operational signs and implied deicmal points are not considered editing symbols.

4. When the GIVING option is not used, a literal must not be specified as the minuend.

5. When dealing with multiple subtrahends, the effect of the subtraction will be as if the subtrahends were first added, and the sum was then subtracted from the minuend.

6. When the CORRESPONDING option is used, treatment of the data-names is similar to that in the CORRESPONDING option of the MOVE statement except that the quantities are added rather than moved. See Notes 6 through 10 in the section on the MOVE verb.

USE

Function

To specify that procedures for input/output error and/or label handling are to be used in addition to procedures supplied by the 7090/7094 Input/Output Control System.

Option 1:

```
USE AFTER STANDARD ERROR PROCEDURE
  ON { INPUT                                  }
     { file-name-1  [, file-name-2...]        }
```

Option 2:

```
USE { BEFORE } STANDARD [ { BEGINNING } ]
    { AFTER  }           { ENDING    }
  [ { REEL } ] LABEL PROCEDURE
  [ { FILE } ]
        ON { INPUT                                  }
           { OUTPUT                                 }
           { file-name-1  [, file-name-2...]        }
```

Notes:

1. A USE sentence, when present, must immediately follow a section header within the Declaratives portion of the Procedure Division. Each USE sentence must be followed by one or more procedure paragraphs. The presence of another section or the statement END DECLARATIVES terminates a USE section.

2. The procedure paragraphs that follow a USE sentence may not include statements containing input/output verbs. They may not include statements which transfer control to procedures containing input/output verbs.

3. When option 1 is specified, the procedure paragraphs which follow the USE sentence will be executed after the standard IOCS error routine is completed. Option 1 may only be used in connection with the input files.

4. When option 2 is specified, the procedure paragraphs which follow the USE sentence will be executed before or after the IOCS header or trailer label checking procedure is completed, depending on whether the BEFORE or AFTER option is specified. Option 2 may only be specified for labeled files.

5. When the BEGINNING option is specified, the procedure paragraphs which follow the USE sentence will be executed for header labels: when the ENDING option is specified, the procedure paragraphs will be executed for trailer labels. If neither BEGINNING nor ENDING is specified, the procedure paragraphs will be executed for both header and trailer labels.

6. When the REEL option is specified, the procedure paragraphs are executed for labels between the first header and the last trailer label of the file. When the FILE option is specified, the procedure paragraphs are executed for the first header and/or the last trailer label of the file. Specifying neither REEL nor FILE is equivalent to specifying both REEL and FILE.

7. When INPUT is specified, the procedure paragraphs are executed for all labeled input files; when OUTPUT is specified, the procedure paragraphs are executed for all labeled output files.

8. When option 2 is specified, external names are generated by the Compiler and placed in the object program Control Dictionary. In jobs consisting of more than one source program, it may, therefore, be necessary to rename non-unique external names by means of $NAME cards. A further discussion of $NAME cards is contained in the publication, IBM 7090/7094 Programming Systems: IBJOB Processor, Form C28-6275-2.

IBM Technical Newsletter

File Number	7090-24
Re: Form No.	J28-6260-2
This Newsletter No.	N28-0124
Date	September 18, 1964
Previous Newsletter Nos.	N28-0103

IBM 7090/7094 COBOL

This newsletter supplements the publication IBM 7090/7094 Programming Systems: COBOL Language: Preliminary Specifications, Form J28-6260-2.

Form J28-6260-1 with Technical Newsletters N28-0070, N28-0092, and N28-0103 is equivalent in information to Form J28-6260-2 with Technical Newsletter N28-0103. The pagination in this newsletter refers to Form J28-6260-2.

In the subject publication, replace the pages listed below with the pages that are attached to this newsletter.

1. Pages 85 and 86
2. Pages 89 and 90
3. Pages 93 and 94
4. Pages 95 and 96

A vertical line immediately to the left of the column shows where the text was changed.

File this cover page at the back of the bulletin. It will provide a reference to changes, a method of determining that all amendments have been received, and a check for determining if the bulletin contains the proper pages.

International Business Machines Corp., Programming Systems Publications, P. O. Box 390, Poughkeepsie, N. Y. 12602

PRINTED IN U.S.A.

DISPLAY

Function

To display low volume data on an available hardware device.

```
DISPLAY  { literal-1  }  [, { literal-2  } ...]
         { data-name-1}     { data-name-2}

[UPON SYSOU1]
```

Notes:

1. The standard display device is the on-line printer. If the UPON option is specified, the display device will be the system output unit. Automatic carriage control for off-line printing of output on the system output unit is provided. For further information concerning SYSOU1, see "FILE-CONTROL" in the Environment Division.

2. When DISPLAY is followed by multiple operands, the data comprising the first operand is displayed as the first set of characters, the data comprising the second operand as the second set of characters, and so on.

3. For any single data item or literal in 7090/7094 COBOL, the maximum number of characters which may be displayed is 72.

4. Internal decimal data items (USAGE COMPUTATIONAL) are prepared for external output with a sign overpunch indicated over the rightmost position. The printer recognizes this as an alphanumeric character and prints it accordingly. The user must interpret this character. Floating point data items are displayed in the scientific decimal form. Other data items are displayed as they appear in core storage.

5. If an output file is assigned to the same device that is used for DISPLAY statements, WRITE statements on that file and DISPLAY statements on that device will not necessarily appear on the listing in the order that the statements of the source program were written. Output resulting from a WRITE statement is buffered whereas output resulting from a DISPLAY statement is produced immediately.

6. The special register TALLY may be used as an operand in a DISPLAY statement.

DIVIDE

Function

To divide one numerical data item into another and set the value of an item equal to the result.

```
DIVIDE { data-name-1 } INTO { data-name-2 } [GIVING data-name-3]
       { literal-1   }      { literal-2   }

    [ROUNDED] [, ON SIZE ERROR { imperative-statement-1 }
                               { NEXT SENTENCE          }

    [{ELSE     } { imperative-statement-2 }]]
     {OTHERWISE} { NEXT SENTENCE          }
```

Notes:

1. The data-names used must refer to the special register, TALLY, or numeric elementary items whose descriptions appear in the Data Division of the program.

2. All rules specified under the verb ADD regarding the size of operands, the ON SIZE ERROR option, the ROUNDED option, the GIVING option, truncation, and the editing of results, apply to the DIVIDE verb. (See ADD.)

3. An error will be indicated at compilation time if the data description for either data-name-1 or data-name-2 specifies the presence of editing symbols. Literals used must be numeric.

4. When the GIVING option is not specified, a literal must not be used as the dividend.

5. Division by zero constitutes a special type of size error. Program control may be provided through the use of a test for zero prior to attempting division. If the zero test type of program control is not provided, the rules specified under the ADD verb with respect to the ON SIZE ERROR option apply.

where A_i is the address of the first word containing X_i and B_i is the number (0 through 5) of the first six-bit group (byte) in the first word containing X_i.

If X_i is a file-name, the corresponding Y_i will be the address of the first word of the File Control Block for the file. A further discussion of File Control Blocks is contained in the publication, IBM 7090/7094 Input/Output Control System, Form C28-6345.

11. The RECEIVE option is used in the subordinate program to move information from the main program to the subordinate program. Data-name-1 through data-name-m following RECEIVE are the names of items into which information is stored. Storing takes place when control is transferred to the subordinate program by means of a CALL statement in the main program.

12. The PROVIDE option is used in the subordinate program to return information to the main program. Data-name-(m+1) through data-name-K following PROVIDE are the names of items whose information is stored into the main program. Storing takes place when control is transferred to the main program by means of option 3.

13. The data-names used with options 1 and 2 must conform to the following rules:

    a. They may not specify items which vary in length at object time.

    b. They may not be report or scientific decimal items.

    c. The total number of data-names specified in the RECEIVE option plus the number of data-names in the PROVIDE option must equal the total number of data-names specified in an associated USING option (that is, where options 1 and 2 have the same 'entry-name'). Any repetitions of data-names must be included in the total number.

    d. The names of the data items in the associated USING option do not have to be identical to the names of the data items in the RECEIVE and PROVIDE options. However, a data item in the USING clause must be of the same length and description as its counterpart in the RECEIVE or PROVIDE option.

    e. The order of the data items in the associated USING option must correspond to the order of its counterparts in the RECEIVE and PROVIDE options.

14. Option 3 is used in the subordinate program to return to the main program. 'Entry-name' in the RETURN option must be the same as 'entry-name' in the CALL statement in the main program.

15. When the DEPENDING ON option is specified, the value of data-name is equivalent to the number of an alternate return specified

in the RETURNING option of an associated CALL statement in the main program. Thus, if the value of data-name is zero, the normal return will be taken; if the value of data-name is 1, return will be to procedure-name-1 specified in the RETURNING option.

16. Following is a coding example illustrating the use of option 1 in a main COBOL program and option 2 and option 3 in a subordinate COBOL program.

Main Program

```
PARAGRAPH-NAME-1.  ENTER LINKAGE-MODE.
     CALL 'ENTPNT' USING WORK, WORK, AMOUNT.
PARAGRAPH-NAME-2.  ENTER COBOL.
```

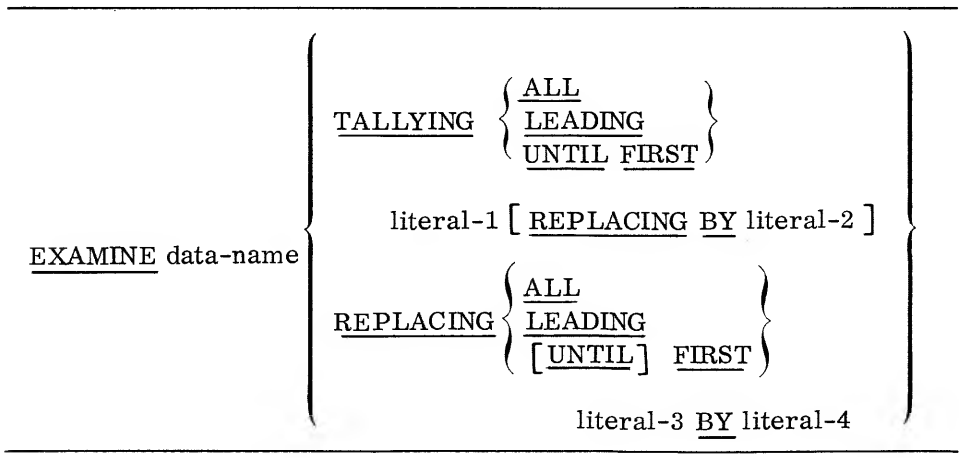
Subordinate Program

```
PARAGRAPH-NAME-3.  ENTER LINKAGE-MODE.
     ENTRY POINT IS 'ENTPNT'
     RECEIVE WORK-A
     PROVIDE WORK-A, AMOUNT-A.
PARAGRAPH-NAME-4.  ENTER COBOL.
                   :
                   :
PARAGRAPH-NAME-5.  ENTER LINKAGE-MODE.
     RETURN VIA 'ENTPNT'.
PARAGRAPH-NAME-6.  ENTER COBOL.
```

EXAMINE

Function

To replace certain occurrences of a specified character and/or to count the number of such occurrences in a data item.



Notes:

1. When using option 1, if the GO TO statement is to be modified by the ALTER verb:
   a. The GO TO statement must itself have a paragraph-name.
   b. The paragraph in which the GO TO statement is included must consist solely of the GO TO statement.

   The paragraph-name assigned to the GO TO statement is referred to by using ALTER verb in order to modify the sequence of the program. If procedure-name-1 is omitted, and if the GO TO statement is not referred to by an ALTER statement prior to the first execution of the GO TO statement, execution of the program will be terminated and control will be returned to IBJOB.

2. In option 2, the contents of data-name must have a positive integral value at object time. The branch will be to the 1st, 2nd, ..., nth procedure-name, as the value of data-name is 1, 2, ..., n. If the value of data-name is anything other than the integers 1, 2, ..., n, then no transfer is executed and control passes to the next statement in the normal sequence for execution.

MOVE

Function

To transfer data, in accordance with the rules of editing, to one or more data areas.

Option 1:

```
MOVE {data-name-1} TO data-name-2 [, data-name-3...]
     {literal     }
```

Option 2:

```
MOVE CORRESPONDING data-name-1 TO
   data-name-2 [, data-name-3...]
```

Notes:

1. Additional receiving areas may be given, following data-name-2. The data designated by the literal or data-name-1 will be moved first to data-name-2, then to data-name-3, etc. When data-name-2 is referred to in this discussion, the note also applies to the other receiving areas.

2. It is improper to use MOVE (without the CORRESPONDING option) for a group item whose format is such that editing would be required on the elementary items in separate operations. If this type of procedure is desired, the CORRESPONDING option must be used, or else each elementary item must be handled individually by means of the verb, MOVE.

3. At object time, data is stored in conformity with the description of the receiving area. When the sizes of the areas of two group items involved in a move are not the same, a warning will be given by the compiler during compilation. A warning is also given for a move from the elementary to a group item or vice versa.

4. When numeric items are moved, they are subject to the following procedures:

   a. When numeric items are moved to numeric fields, they are aligned by decimal points with zero filling or truncation on either end as required. A warning message is given by the compiler if there is a possibility that significant digits will be lost through truncation when the program is executed.

   b. When internal decimal items (COMPUTATIONAL) are moved to external decimal fields (DISPLAY), they are converted from binary to BCD; i.e., USAGE becomes DISPLAY. Since COMPUTATIONAL items are assumed to be signed, the DISPLAY PICTURE will be preceded by an S.

   c. When numeric items are moved to report fields for editing, they are aligned as specified in the report field. (Editing is explained in the report form of the PICTURE clause.) After the item is edited, it is treated as alphanumeric data when referred to in the program.

   d. Numeric items are stored in alphanumeric fields from left to right and the signs are dropped. Non-significant digits are truncated if the item is too long. Trailing blanks fill in the alphanumeric field if the item is too short. The following examples show moves from numeric to alphanumeric fields (b represents a blank):

Source Field Picture	Initial Source Contents	Receiving Field Picture	Resulting Receiving Field Contents
X(4)	1.23	X(6)	1.23bb
9V99	1.23	X(6)	123bbb
999	123	X(6)	123bbb
S9(6)	-000199	X(4)	0001
9(4)V9(5)	1234.56789	X(72)	123456789bb...b

   The last example shows that there is no difference between treatment of alphanumeric items of lengths greater than or not greater than 18.

5. When non-numeric items are moved:

   a. The characters are placed in the receiving area left to right.

   b. If the source field is shorter than the receiving field, the remaining character positions are filled with spaces.

   c. If the source field is longer than the receiving field, the generated code provides for termination of the move as soon as the receiving field has been filled. A warning message is given for this condition unless one has already been given as stated in note 3.

   d. A table of legal moves, using the verb MOVE, is given below.

      A detailed description of the types of fields represented may be found under the PICTURE clause in the Data Division. Numbers in parentheses in the table refer to subsequent notes.

	Receiving Field Type							
Source Field Type	Group Item (1)	Alpha-betic	Alpha-numeric	Report	External Decimal	Internal Decimal	Floating Point	Scientific Decimal
Group Item (1)	yes	yes	yes	no	no	no	no	no
Alphabetic	yes	yes	yes	no	no	no	no	no
Alphanumeric	yes	yes	yes	no	no	no	no	no
Report	yes	no	yes	no	no	no	no	no
External Decimal	yes	no	yes	yes	yes	yes	yes	yes
Internal Decimal	yes	no	yes	yes	yes	yes	yes	yes
Floating Point	yes	no	yes	yes	yes	yes	yes	yes
Scientific Decimal	yes	no	yes	yes	yes	yes	yes	yes
ZERO [S] or ZEROES	yes	no	yes	yes(2)	yes	yes	yes	yes(2)
SPACE [S]	yes	yes	yes	yes(3)	nc	no	no	no
LOW-VALUE [S]	yes	no	yes	no	no	no	no	no
HIGH-VALUE [S]	yes	no	yes	no	no	no	no	no
QUOTE [S]	yes	no	yes	no	no	no	no	no
ALL---	yes	yes(4)	yes	no	yes(5)	no	no	no

(1) Group items are treated as having a PICTURE of all X's.

(2) The value zero is moved in accordance with editing requirements.

(3) A warning message is given.

(4) The character must be alphabetic.

(5) The character must be numeric.

e. For source fields of the scientific decimal type:

At object time a free form of data is allowed within the limits of the field. For example, a field with PICTURE -99V9E-99 may contain the value 1 in any of the following ways (where b represents a space):

b . 01b2b	(01×10^2)
1bb+01b	(note scale applied when no point)
.001bb3	
b10bbbb	(note scale applied when no point)
1000.-3	
etc.	

Note that the letter E is never part of the data.

6. If the CORRESPONDING option is used, selected items within data-name-1 are moved, with any required editing, to selected areas within data-name-2. Items are selected by matching the data-names of areas defined within data-name-1 with like data-names of areas defined within data-name-2, according to the following rules:

   a. At least one of the items of a selected pair must be an elementary item.

   b. The respective data-names are the same including all qualification up to but not including data-name-1 and data-name-2.

   Each CORRESPONDING source item is moved in conformity with the description of the receiving area. The results are the same as if the user had referred to each pair of CORRESPONDING data-names in separate MOVE statements.

7. If an area used as either a source or receiving field is described with a REDEFINES clause, the MOVE CORRESPONDING statement refers to the redefined area, not the original description.

8. If the CORRESPONDING option is used, no items in the group referred to can contain an OCCURS clause.